VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

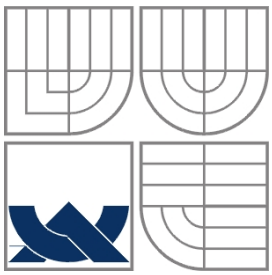
VÝUKOVÝ PROGRAM PRO DEMONSTRACI PRINCIPU BAREV A BAREVNÝCH MODELŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

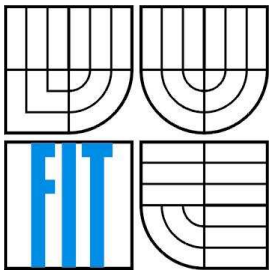
AUTOR PRÁCE
AUTHOR

JAROMÍR VOJÍŘ

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÝUKOVÝ PROGRAM PRO DEMONSTRACI PRINCIPU BAREV A BAREVNÝCH MODELŮ

EDUCATION COMPUTER PROGRAM FOR DEMONSTRATION OF COLORS AND COLORS MODELS PRINCIPLES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAROMÍR VOJÍŘ
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. PŘEMYSL KRŠEK, PH.D.
SUPERVISOR

BRNO 2009

Zadání bakalářské práce/8865/2008/xvojir01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2008/2009

Zadání bakalářské práce

Řešitel: **Vojíř Jaromír**
Obor: Informační technologie
Téma: **Vyukový program pro demonstraci principu barev a barevných modelů**
Kategorie: Počítačová grafika

Pokyny:

1. Prostudujte problematiku definice barev v počítačové grafice a barevných modelů
2. Prostudujte problematiku tvorby výukových a demonstračních programů
3. Navrhněte výukový program pro demonstraci principu barev v počítačové grafice a barevných modelů
4. Implementujte navržený program ve vybraném jazyce (C/C++, Java, Python, C#)
5. Zhodnoťte dosažené výsledky a stanovte další vývoj projektu

Literatura:

- Žara J., Beneš B., Felkel P.: Moderní počítačová grafika. 1. vyd. Praha, Computer press 1998, 448 s., ISBN 80-7226-049-9

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních 3 bodů zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kršek Přemysl, doc. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2008
Datum odevzdání: 20. května 2009

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Práce se zabývá vývojem výukového programu pro demonstraci principů barev a barevných modelů. Obsahem práce je teoretický rozbor dané problematiky zahrnující oblast od základních pojmů až po využití jednotlivých barevných modelů. Následuje podrobná analýza požadavků pro vývoj daného výukového programu. V části implementace je pak popsán proces samotného vývoje aplikace. Na závěr je diskutován přínos práce a možnost budoucích úprav. Program by mohl sloužit jako doplněk studijních materiálů pro studenty.

Abstract

The thesis deals with development of an educational program for demonstration of principles of colors and of color models. The thesis contains theoretical study of the given question from basic terms up to use of individual color models. This is followed by a detailed analysis of the requirements necessary for development of the given educational program. In the implementation part, the process of development of the application itself is described. In the end of the thesis there is a discussion of contribution of the thesis and of possibility of future adaptations. The program can serve as a complement of study materials for students.

Klíčová slova

výukový počítačový program, světlo, pigment, barva, barevné modely

Keywords

educational computer program, light, pigment, color, color models

Citace

Vojíř Jaromír: Výukový program pro demonstraci principu barev a barevných, bakalářská práce, Brno, FIT VUT v Brně, 2009

Výukový program pro demonstraci principu barev a barevných modelů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. Ing. Přemysla Krška, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jaromír Vojíř
20. 5. 2009

Poděkování

Rád bych poděkoval vedoucímu práce Doc. Ing. Přemyslu Krškovi, Ph.D. za ochotu, vstřícnost a odborné rady poskytnuté při tvorbě této práce.

Obsah

1 Úvod

Problematika barev a barevných modelů je nedílnou součástí základních znalostí o počítačové grafice. Barva je jedním ze základních atributů obrazu. Pro každý bod obrazu definujeme barvu, bez ohledu na to zda jde o grafiku rastrovou nebo vektorovou. Pro mnoho lidí je stále neuvěřitelné, že všechny barvy, se kterými počítače pracují, jsou jen kombinací několika základních barev. Studentům jsou většinou známé barevné modely RGB a CMY. Barevný prostor RGB je vhodný zejména pro displeje a podobná zobrazovací zařízení. Jeho skládání barev však naprosto neodpovídá lidským zkušenostem s mícháním barev. Běžným lidským znalostem, kde každým přidáním barevného pigmentu barva tmavne, odpovídá barevný prostor CMY. Ten je vhodný zejména pro tiskařské techniky. Další barevné prostory, které jsou lidskému popisu barev daleko bližší, již tak známé nejsou.

Cílem této bakalářské práce je seznámit demonstrační a zábavnou formou studenty s těmito barevnými modely (HSV, HSL) a ukázat propojení s těmi, které už znají. Teoretický základ k problematice barev a barevných modelů je zpracován v kapitole 2. Alespoň základní znalosti této problematiky jsou nutné k plnému využití výukové aplikace a z tohoto důvodu bylo rozhodnuto začlenit výukovou část do aplikace. O jejím návrhu pojednává kapitola 3. Zde jsou také rozebrány návrhy dalších částí aplikace, jako je část testová a část pro práci s modely. Implementace a použité technologie v aplikaci se nachází v kapitole 4. Kapitola 5 shrnuje výsledky práce a ukázky ovládání aplikace. Závěrečná kapitola 6 pak pojednává o možných rozšířeních aplikace a poznatcích, ke kterým se během psaní bakalářské práce dospělo.

2 Teorie barev a barevných modelů

Tato kapitola popisuje problematiku barev a barevných modelů. Nejprve se zaměříme na světlo a barvu samotnou. Podkapitola 2.1 shrnuje základní pojmy a objasňuje fyzikální vlastnosti viditelného EM spektra.

2.1 Světlo a barvy

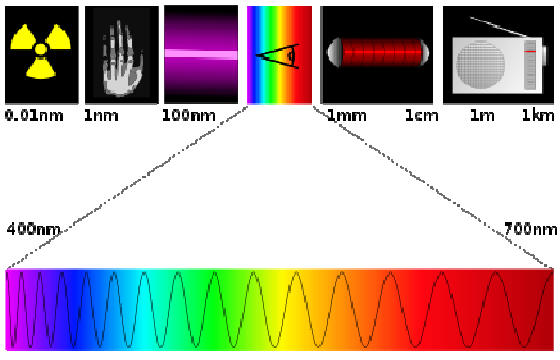
Tato podkapitola se pokusí nastínit problematiku barev a světla v teoretické rovině. Nejprve se sluší uvést základní definice, ze kterých se bude v následujícím textu vycházet.

Světlo je elektromagnetické záření o vlnové délce viditelné okem, obecněji elektromagnetické vlnění v rozmezí od infračerveného po ultrafialové. V některých oblastech vědy a techniky může být světlem chápáno i elektromagnetické záření libovolné vlnové délky. Tři základní vlastnosti světla (a elektromagnetického vlnění vůbec) jsou svítivost (amplituda), barva (frekvence) a polarizace (úhel vlnění). Kvůli dualitě částice a vlnění má světlo vlastnosti jak vlnění, tak částice. Studiem světla a jeho interakcemi s hmotou se zabývá optika.[1]

Barva je vjem, který vytváří viditelné světlo dopadající na sítnici lidského oka. Barevné vidění lidského oka zprostředkují receptory, zvané čípky, trojího druhu – citlivé na tři základní barvy: červenou, zelenou a modrou.[2]

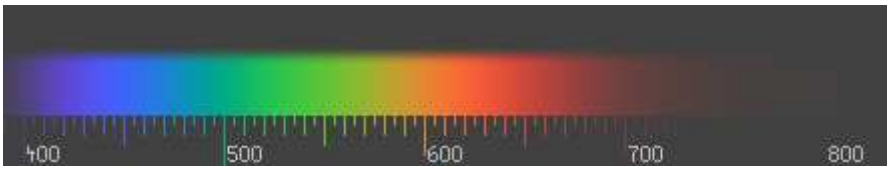
2.1.1 Fyzikální vlastnosti světla

Nejdůležitější fyzikální veličinou, která charakterizuje elektromagnetické záření, je jeho vlnová délka. Podle vlnové délky ve vakuu, popř. frekvence elektromagnetického vlnění rozlišujeme několik druhů elektromagnetického záření, jejichž souhrn vytváří spektrum elektromagnetického záření. Přehledně jsou všechny druhy elektromagnetického záření vyznačeny na obrázku 2.1. Mezi jednotlivými druhy elektromagnetického záření není ostrá hranice, přechody mezi nimi jsou plynulé nebo se oblasti záření i překrývají.



Obrázek 2.1 - EM spektrum[3]

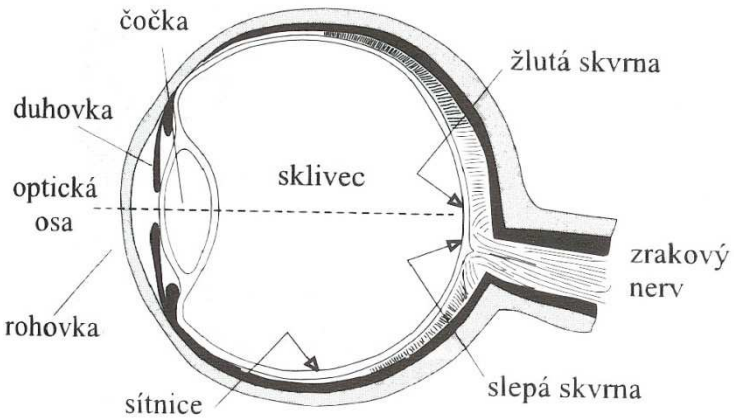
Viditelná část elektromagnetického (EM) spektra, nalézajícího se v oblasti vlnových délek 380-720nm, je lidmi vnímána jako světlo. Barva je pak pro nás záření s vlnovými délkami uvnitř této oblasti. Při spodní hranici 380nm se nachází barva fialová, která později přechází v neviditelné záření ultrafialové. Na opačném konci, při hranici 720nm, máme barvu červenou. Za touto hranicí přechází v infračervené záření. Rozložení barev ve spektru se nachází na obrázku 2.2



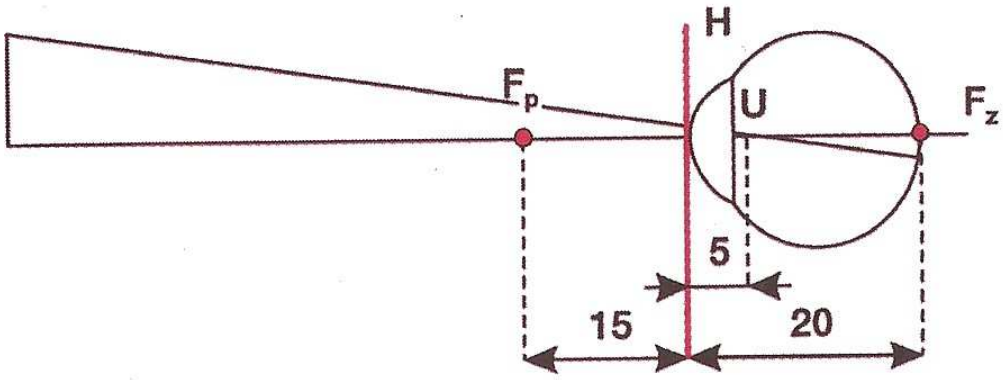
Obrázek 2.2 - Viditelné barevné spektrum

2.1.2 Lidské oko

Lidské oko (obr 2.3) se blíží tvarem kouli o průměru přibližně 20mm. Z lékařského hlediska je oko složitý optický systém (rohovka, komorová voda, čočka, sklivec). Jeho hlavní bod leží na přímce (optická osa), procházející předním předmětovým (Fp), zadním obrazovým (Fz) ohniskem a středem kulové plochy (rohovky), v místě přechodu centrálního paprsku z jednoho prostředí do druhého. V hlavním bodu stojí kolmo na optickou osu hlavní rovina H. Paprsek z horního okraje předmětu prochází uzlovým bodem U, tj. optickým středem, aniž se láme (obr. 2.4). Zraková osa se odchyluje od optické osy asi o 5° a prochází žlutou skvrnou. Lomivost čočky v dioptriích (D) se stanoví z převrácené hodnoty ohniskové vzdálenosti v metrech. Lomivost 1D odpovídá ohniskové vzdálenosti 1m, 2D 0,5m atd. Protože přední ohnisko leží asi 17mm před okem, celková lomivost oka je 1/0,017m, tj. necelých 59D. Rohovka se tu podílí více než čočka, jejíž lomivost se však může změnit akomodací, v dětství až o 16D.[4]



Obrázek 2.3 - Lidské oko [5]



Obrázek 2.4 - Redukované oko [4]

Naše sítnice obsahuje dva druhy receptorů – tyčinky a čípky. Tyčinek máme přibližně 15 krát tolik než čípků. Oba dva druhy receptorů se však na sítnici vyskytují v počtu milionů. Tyčinky nám slouží hlavně pro noční vidění, protože jsou přibližně 10x citlivější na světlo než čípky. Naopak čípky jsou základem našeho barevného vidění. Každý čípek v sobě obsahuje jeden ze tří druhů fotopigmentů (červený, zelený, modrý).

Jeden fotopigment má nejvyšší citlivost v oblasti krátkých vlnových délek viditelného spektra okolo 445nm a je necitlivý na vlnové délky delší než 520nm. Obvykle je označován jako modrý fotopigment. Další dva typy fotopigmentů jsou nejcitlivější na 535nm a 575nm, ale reagují prakticky na všechny vlnové délky ve viditelném spektru. Jsou označovány jako zelené a červené fotopigmenty. Pojmenování není odvozeno od jejich barvy, ale z barevného vjemu, který je spojen s jejich maximální citlivostí. Navíc název červeného fotopigmentu není zcela přesný, neboť jeho citlivost je největší v oblasti žlutého světla. Barevné pigmenty nejsou zastoupeny na sítnici ve stejném poměru. Přibližně 64% čípků obsahuje červené a 32% zelené pigmenty. Modré pigmenty jsou obsaženy asi ve 2% čípků.[5]

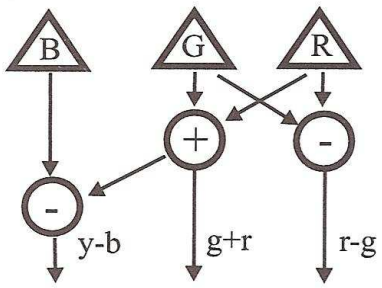
Tyčinky a čípky nejsou rozmístěny rovnoměrně po celé sítnici. Čípky jsou soustředěny na malé ploše, přibližně 1,5mm, umístěné v průsečíku optické osy oka se sítnicí. Tato oblast, nazývaná žlutá skvrna, obsahuje okolo 300 000 čípků. Vynikající zrak dravých ptáků je částečně vysvětlován také tím, že mají v oblasti žluté skvrny čtyřikrát více fotoreceptorů než lidé.[5]

2.1.3 Vnímání barev a jasu

Zrakový nerv je tvořen svazkem nervových vláken, která předávají signály od fotoreceptorů do mozku. Na této nervové cestě dochází k rekombinaci barevné informace.

Ze tří původních informačních kanálů r, g, b vznikají po rekombinaci tři odlišné kanály (obr 2.5): jeden kanál nese informaci o poměru červená-zelená (r-g), druhý kanál o poměru žlutá-modrá (y-b). Tyto kanály poskytují informace o barvě, zatímco třetí kanál zelená+červená (g+r) indikuje jas. Jelikož mozek vyhodnocuje barvu až na základě rekombinovaných signálů, plynou z této

transformace zajímavé důsledky. Faktem je, že lidé nejsou schopni vnímat některé barevné kombinace současně. Můžeme si položit otázku, zda jsme někdy viděli nažloutlou modř nebo nazelenalou červeň. Protože se oko zaostřuje na hrany podle výrazných změn jasu, plyne z toho, že hrany a tvary nejsou téměř rozlišitelné v odstínech modré barvy.[5]



Obrázek 2.5 - rekombinace barevných stimulů [5]

2.2 Barevné modely

Základním principem fungování barevných modelů je míchání barev. Různé barvy, které se používají při vytváření obrazu jsou jen kombinací několika základních barev. V případě modelu založeném na světle, tedy RGB, je kombinací všech složek v plné intenzitě bílá barva. Naopak u CMY modelu založeném na pigmentu výsledná barva postupným přidáváním složek tmavne. V plné intenzitě pak tvoří barvu černou. Ostatní barevné modely si kladou za cíl umožnit přirozenější a intuitivnější míchání barev uživateli.

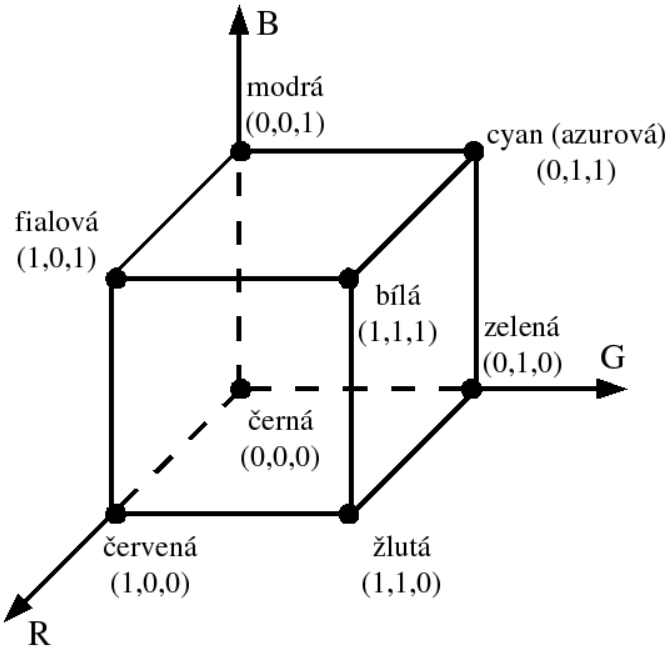
2.2.1 Modely RGB a RGB*α*

RGB barevný model je založen na teorii Younga–Helmholtze, trojbarevného vidění, a na Maxwellově barevném trojúhelníku. Použití RGB barevného modelu, jako standardu pro prezentaci barev na internetu, má své kořeny v letech 1953 RCA barevné TV normy a v použití Edwin Loandova RGB standardu v Land/Polaroidu.[6]

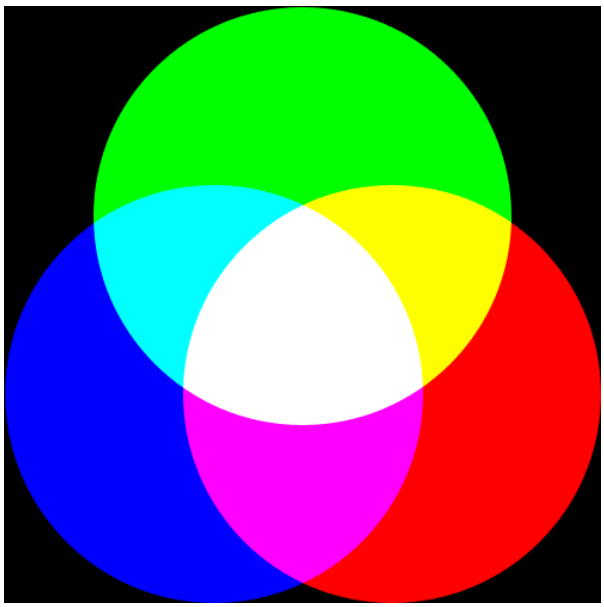
RGB

Význam zkratky RGB je prostý. Každé písmeno značí anglický název barvy – Red, Green a Blue. Jsou to základní složky tohoto aditivního barevného modelu. Aditivní proto, že smícháním všech tří složek – červené, zelené a modré v odpovídající intenzitě získáme bílou barvu. Nejběžnější příklad použití tohoto modelu je u obrazovek, ať už CRT nebo LCD panelů, či jiných zobrazovacích zařízení. Každý pixel na obrazovce je reprezentován v grafické kartě pomocí třech osmibitových hodnot,

reprezentujících intenzitu jednotlivých složek. Použitím různých kombinací intenzit červené, zelené a modré, můžeme vytvořit velmi mnoho barev. Přesně vyjádřeno je to kombinace 256^3, což představuje pro někoho neuvěřitelných 16 777 216 různých barev.



Obrázek 2.6 - Model RGB [6]



Obrázek 2.7 - Aditivní míchání barev [6]

R	G	B	Název barvy
0	0	0	**černá**
255	255	255	**bílá**
255	0	0	**červená**
0	255	0	**zelená**
0	0	255	**modrá**
255	255	0	**žlutá**
255	0	255	**purpurová**

Tabulka 2.8 - základní barvy

Dalším častým místem k setkání s RGB modelem je webdesing. Pomocí RGB určujeme, např. v kódu HTML nebo CSS, jaké barvy má použít prohlížeč například pro pozadí, nebo jakou má zvolit barvu písma. Protože zapamatovat si barevné kombinace RGB složek je obtížné, často se pro toto použití základní barvy reprezentují jako jejich slovní vyjádření. Místo (0, 255, 255) tedy použijeme raději název barvy.

Jak již bylo naznačeno výše, složením barevných složek v plné intenzitě získáme barvu bílou. Podobně, pokud snižujeme intenzitu u všech složek stejně, získáme odstíny šedé. Minimální, tedy nulová intenzita všech barevných složek, nám dá barvu černou. Takto získáme 255 odstínů šedi. Když však budeme převádět barevný obrázek na černobílý, nemůžeme pouze sečíst hodnoty barevných složek, podělit třemi a rovnoměrně je rozdělit mezi barevné složky. Naše oči totiž nevnímají intenzitu jednotlivých barevných složek stejně.
Pro převod využíváme empiricky odvozený vztah (viz.vzorec 2.9), který nám definuje rozložení vnímání barev na jednotlivé složky. [5]

$$I = 0.299\,R + 0.587\,G + 0.114\,B$$

Vzorec 2.9 - rozložení vnímání barev

RGBA

V počítačové grafice se setkáváme s další zkratkou, která připomíná prostor RGB. Jedná se o kombinaci RGBA. Tato zkratka je používána pro vyjádření skutečnosti, že barevný obraz zapsaný v prostoru RGB je doplněn informací o průhlednosti. Každý barevný bod takového obrazu s sebou

nese skalární údaj, který určuje, v jakém rozsahu pokrývá barva plochu obrazového bodu. Hodnota 0.0 znamená neprůhledný bod, maximální hodnota 1.0 pak bod zcela průhledný.[5]

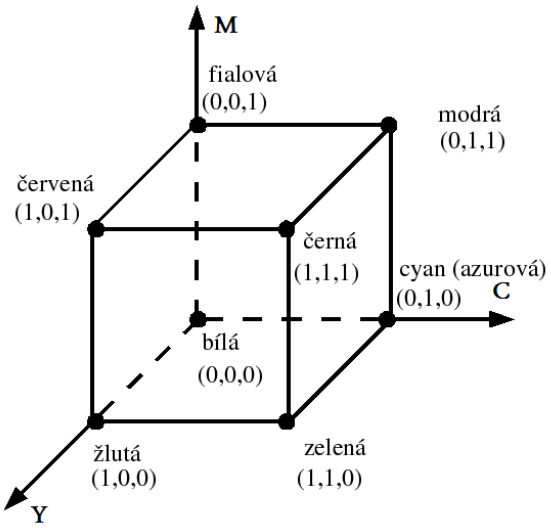
RGBA tedy zahrnuje jakýsi extra 8bitový kanál pro průhlednost. Toto má za následek jeho 32bitový formát. Tento formát lépe odpovídá současnému hardware, který efektivněji a rychleji pracuje s násobky dvou, než s jakýmkoliv jiným číslem způsobujícím přesah adres.

2.2.2 Modely CMY a CMYK

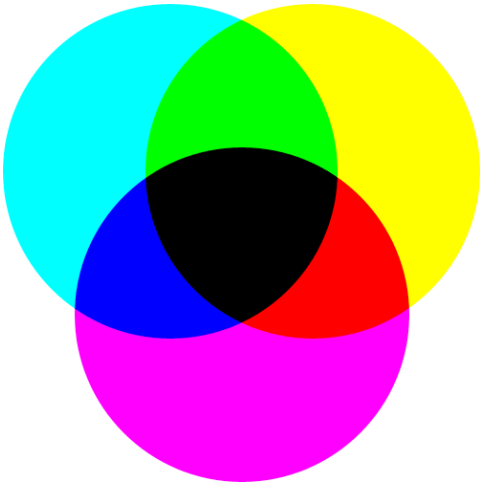
Barevný model založený na subtrakci neboli odečítání barev, které se odráží od povrchu a jehož nejběžnějším příkladem je tiskárna, která na papíře tvoří určitý barevný pigment. Tak by se ve zkratce dal popsat model CMY. Model CMY obsahuje tři základní barvy – Cyan (azurová), Mangenta (purpurová), Yellow (žlutá). Převod mezi tímto modelem a aditivním modelem RGB je prostý. Jeden model je doplňkem druhého viz vzorec (2.10) a obrázek (2.11, 2.12).

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Vzorec 2.10 - vztah RGB - CMY



Obrázek 2.11 - Model CMY [7]



Obrázek 2.12 - Subtraktivní míchání barev [7]

Nejdříve je potřeba si ujasnit, proč v modelu RGB máme červenou, zelenou a modrou složku, zatímco v modelu CMY máme azurovou, purpurovou a žlutou. Nejnázornější to bude ukázat například na oblíbené i nenáviděné žluté barvě. Žluté světlo v modelu RGB vznikne kombinací červené a zelené, tedy chybí mu modrá část spektra. Žlutý pigment – například žlutě pomalovaný papír, proto právě tuto modrou barvu pohlcuje. Když následně na náš žlutý papír dopadne bílé světlo, tak odrazí zpět pouze kombinaci červené a zelené složky. A kombinací červené a zelené je, jak víme, žlutá. Proto předmět pokrytý žlutým pigmentem vidíme žlutě.

Zatímco RGB je model, který se uplatní u zobrazovacích zařízení, využití CMY modelu je nám daleko bližší už od školních let. Při malování nanášíme jednu barvu na druhou a s každým přidaným odstínem, např. barvičkou na paletě „vodovek“, náš výsledný výtvor tmavne. Stejně tedy pracuje i model CMY. Při nulové intenzitě všech tří složek máme papír – barvu bílou, pokud však použijeme všechny složky v maximální intenzitě, dostaneme barvu černou – resp. barvu jí připomínající. Proč nedostaneme vždy černou barvu, si povíme dále.

V teoretické rovině nám tedy postačují jen tři složky. Problémem je, že kombinací všech složek na maximální úrovni, by měla vzniknout černá barva. Ve skutečnosti při aplikaci tohoto modelu v reprodukčních zařízeních, jako je tiskárna, je produktem této kombinace pouze tmavě šedá barva. Nehledě na to, tisknout na papír černou tak, že použijeme všechny barvy, je poněkud neekonomické. Tento problém řeší zavedení další složky a to složky K, kde K značí černou barvu, tedy „blacK“. Zavedením této složky, například pro použití v tiskárnách, získáme určité výhody. Jedná se hlavně o levnější tisk – černý inkoust lze vyrobit levnější než tří-barevnou cartrige, dále pak rychlejší tisk – barva nám dříve zaschne. Asi nejdůležitější je však fakt, že většina textu je tištěna pouze černou barvou. Získání opravdu černé barvy, jen pomocí CMY složek, by vyžadovalo značné úsilí a perfektní barevný soutisk.

Problémy s převodem RGB na CMYK

Obrázky či fotografie zpracovávané na počítači je potřeba před tiskem převést z modelu RGB na model CMYK. Fotka na monitoru je totiž zobrazena modelem RGB, zatímco naše tiskárna tiskne podle modelu CMYK. O tento proces se často stará ovladač tiskárny. Ovšem ani ten kouzlit neumí. O co jde, je patrné z následujících obrázků 2.13 a 2.14. CMYK špatně pokrývá barvy jako sytě červenou, zelenou nebo modrou. Je to proto, že monitor např. červenou barvu přímo vyzařuje, kdežto výtisk červenou barvu jen odráží. Přesto pomocí moderních fototiskáren, je výsledná fotografie skoro stejně barevná jako naše fotka na monitoru.

Obrázek 2.13 - CMYK [7]

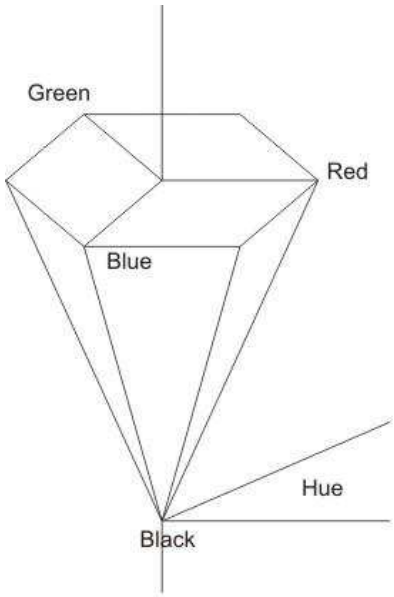
Obrázek 2.14 - RGB [7]

2.2.3 Model HSV a HLS

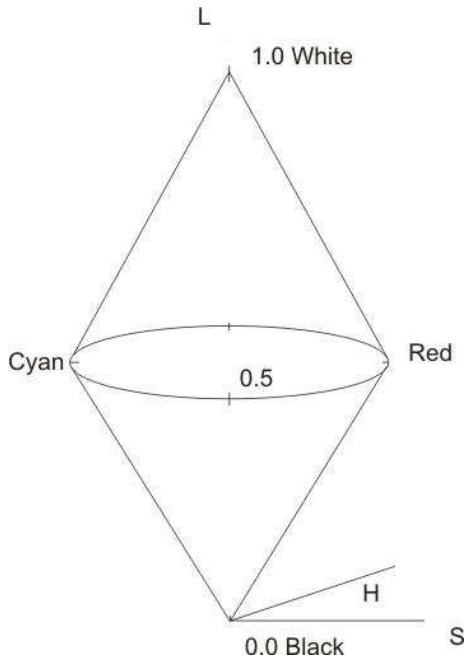
Modely HSV a HLS, které budou popsány v této části, se od výše uvedených liší jednou podstatnou věcí. Jejich základní složky tentokrát netvoří barvy. U modelu HSV je to H (*hue*, barevný tón), S (*saturation*, sytost), V (*value*, jasová hodnota). Pokud se podíváme blíže, co tyto zkratky znamenají, pak zjistíme, že barevný tón nám určuje převládající barvu, sytost nám určuje jak velká je příměs jiných barev a konečně jasová hodnota udává intenzitu bílého světla. Jak je již popsáno výše, hodnotu RGB na HSV a obráceně lze přepočítávat. Tento převod má formu algoritmu a lze jej najít například na webu, nebo v odborné literatuře.

Pro zobrazení prostoru HSV se používá šestibokého jehlanu, (viz. obr 2.15) , kde souřadnice *s* a *v* se mění v rozsahu 0-1 (v programu přepočítáno na 0-100%) a *h* v rozsahu 0-360°. Vrchol jehlanu reprezentuje černou barvu. Střed podstavy pak barvu bílou. Nejjasnější barvy jsou na okrajích podstavy, kde při změně hodnoty úhlu *h* dochází ke změně barevného tónu, nikoli však ke změně saturace nebo jasu. Přesto není tento model dokonalý. Při tomto pohybu po hraně jehlanu zůstává hodnota *s* na maximu, přitom se však nepohybuje přirozeně po kružnici, ale po šestiúhelníku. Tento problém napravuje model HLS.

Model HLS (*h* - odstín, *l* - světlost , *s* - saturace) odstraňuje některé nedostatky modelu HSV. Reprezentuje ho „oboustranný kužel“ (viz. obr 2.16), který nejlépe vystihuje fakt, že lidské oko registruje nejvíce barevných tónů při průměrné světlosti. Nejvíce barev tedy v modelu nalezneme s nastavením *l = 0,5*, kdy zároveň při plné saturaci máme pocit nejjasnější barvy. Více informací je možno nalézt např. na [8].



Obrázek 2.15 HSV



Obrázek 2.16 HLS

Prostory HLS a HSV jsou vhodné zejména pro práci uživatele s barvami a pro vzorníky barev v grafických editorech, je to zejména pro jejich větší přirozenost zadávání barevného tónu. Zadávat dominantní barvu a jas je uživateli daleko bližší, než zadávání kombinace barevných složek v RGB. Při práci s těmito modely je však třeba si dát pozor na některé kombinace hodnot. Pokud například zvolíme saturaci rovnu nule, pak je úplně jedno, jaké hodnoty nabývá barevný tón. Jeho hodnotu pak považujeme za hodnotu nedefinovanou.

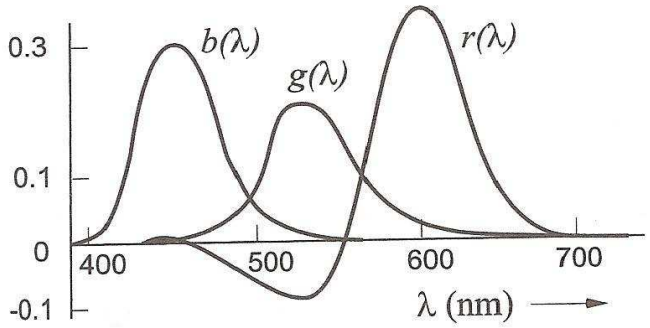
2.2.4 Chromatický diagram CIE

Důležitá nutnost objektivního porovnání, měření a vyjadřování barev, které by nebylo zatížené chybou subjektivního lidského vnímání, vyústilo v roce 1931 vytvořením mezinárodního standardu základních barev. Součást toho standardu je i tzv. chromatický diagram CIE 1931.

Ve standardu se předpokládá, že každou barvu lze definovat váženým součtem tří primárních barev, které jsou vyzařovány monochromatickými zdroji světla o vlnových délkách R = 700nm, G = 541.1nm a B = 435.8nm. Libovolná monochromatická barva ve viditelném spektru je určena vztahem

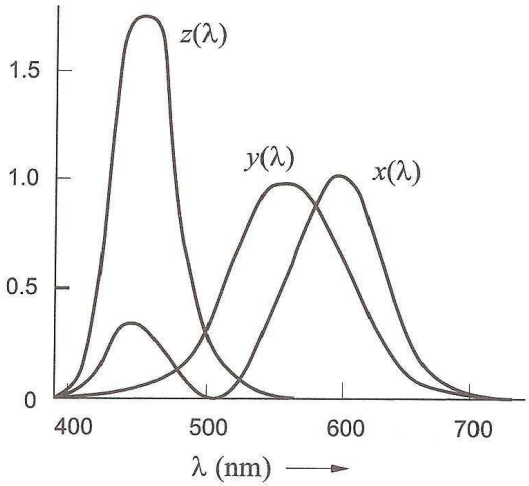
$$C = r.R + g.G + b.B$$

Pomocí kolorimetrických experimentů zaměřených na srovnávání takto vytvořených barev s danými monochromatickými barvami, byly stanoveny barevné srovnávací funkce r(λ), g(λ), b(λ) s průběhy na obrázku 2.17.[5]

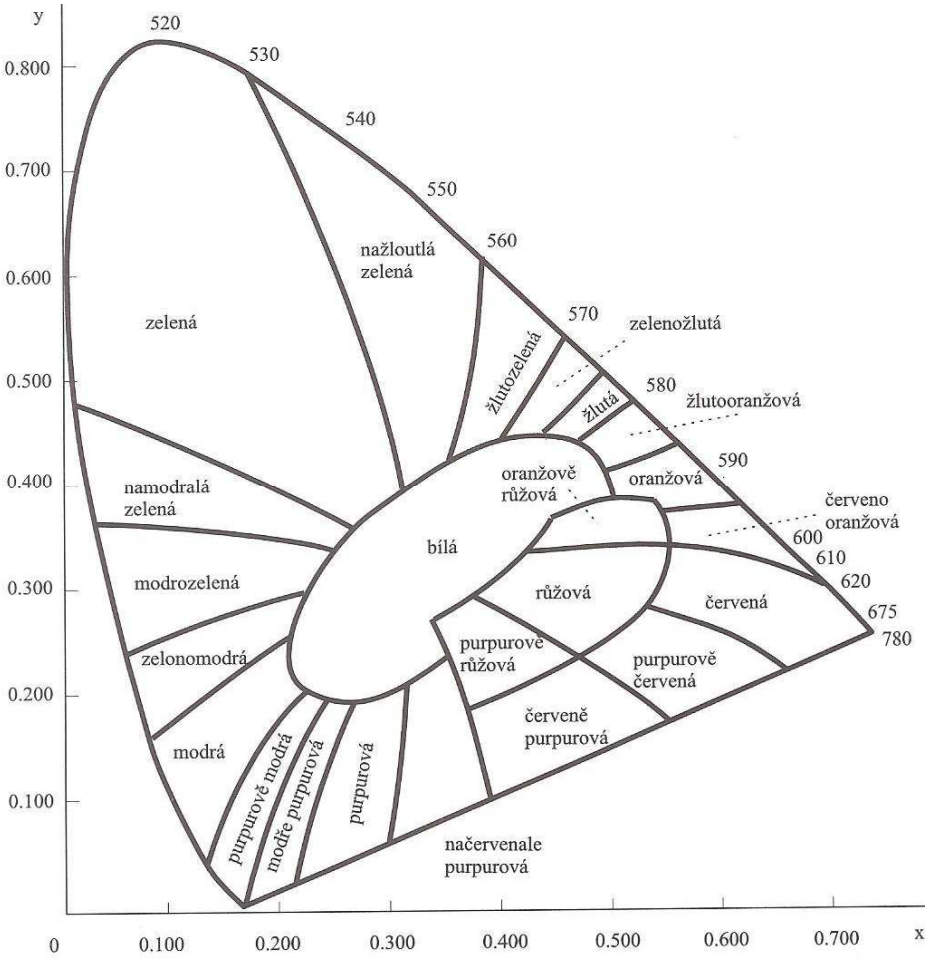


Obrázek 2.17 - Srovnávací funkce RGB [5]

Některé monochromatické barvy však nelze pomocí srovnávacích funkcí r(λ), g(λ), b(λ) vytvořit. Subjektivně stanovená hodnota funkce r(λ) nabývá v části viditelného spektra záporné hodnoty. Namíchané barvy zde mají při porovnání s monochromatickou barvou "přebytek" červené, i když jsou vytvořeny pouze pomocí zelené a modré složky. Proto byly ve standardu CIE 1931 zavedeny umělé barevné srovnávací funkce a průběhy podle obrázku 2.18.[5]



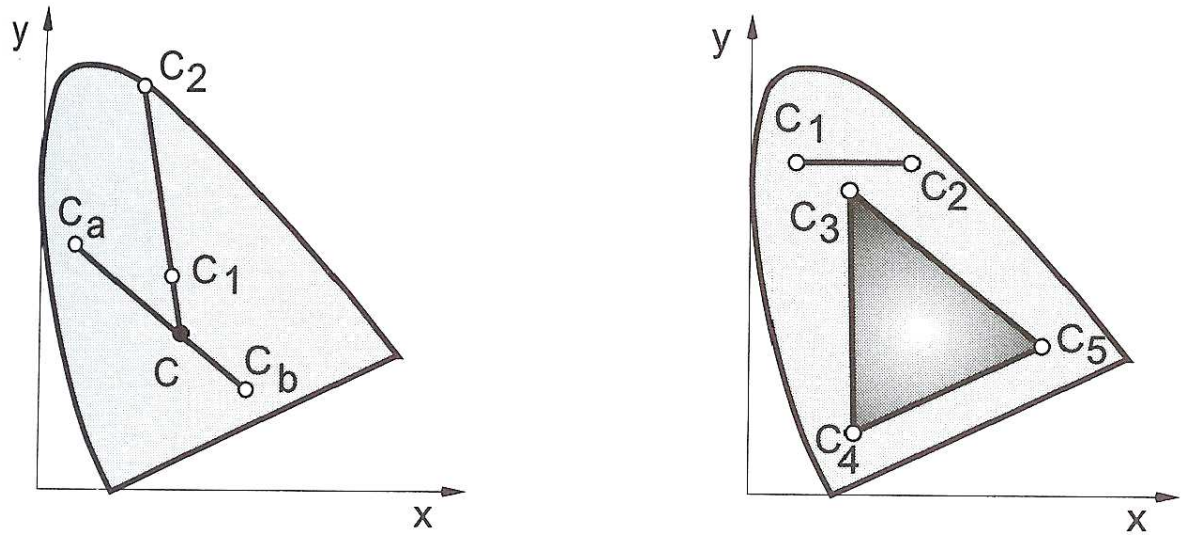
Obrázek 2.18 - Srovnávací funkce XYZ [5]



Obrázek 2.19 - Barevný prostor CIE 1931 [5]

Barevné body tvořící obalovou spektrální křivku jsou označeny vlnovou délkou v nanometrech, počínaje červenou částí spektra a konče fialovou částí spektra. Bod C v diagramu (obr 2.20) odpovídá poloze bílého světla. Chromatický diagram poskytuje prostředky pro kvantitativní určení sytosti a dominantní vlnové délky, stejně jako z něj lze odvodit další barevné veličiny a vztahy.[5]

Pro libovolný barevný bod, jako například C1 na obrázku 2.20 vlevo, definujeme *sytost barvy* jako relativní vzdálenost barevného bodu od standardního bílého světla C. Měření vzdálenosti se provádí na polopřímce spojující body C a C1 a protínající obalovou křivku v bodě C2. Barva C1 na obrázku je sytá asi z 25%, protože leží přibližně v jedné čtvrtině orientované úsečky mezi C a C2. *Dominantní vlnová délka* jakékoliv barvy, je definována jako vlnová délka na spektrální křivce protínající úsečku spojující C a příslušný barevný bod C1. Pro barevný bod C1 na obrázku 2.20 je dominantní vlnová délka v bodě C2. *Doplňkové barvy* jsou reprezentovány libovolnými body úsečky, která prochází bodem C, na obrázku 2.20 vlevo dvojice barev Ca, Cb. Pokud mají dvě doplňkové barvy stejnou sytost, vznikne jejich složením bílé světlo C.[5]



Obrázek 2.20 – CIE 1931: Sytost, dominantní vlnová délka a doplňkové barvy (vlevo), barevné rozsahy (vpravo) [5]

Všechny vnitřní body trojúhelníka C_3-C_4-C_5 (obr 2.20 vpravo), lze vytvořit kombinací základních barev v jeho vrcholech. Takový trojúhelník uvnitř chromatického diagramu se nazývá barevný rozsah (color gamut). Jak vyplývá z geometrie chromatického diagramu, není možno nalézt takové tři základní barvy, které by určovaly trojúhelník pokrývající celý diagram. Toto odpovídá skutečnosti, že z konečného počtu základních barev nelze vytvořit všechny existující barvy. Počítačem generované obrazy proto obsahují méně barev než skutečný svět.[5]

3 Analýza požadavků a návrh aplikace

Dobrá analýza zadané bakalářské práce je jedním ze základních kamenů pro její úspěšné zvládnutí. Práce byla zadána jako výukový program a podle toho k ní přistupujeme. Je třeba zajistit, aby bylo možno se o problematice něco dozvědět, poznatky si prakticky vyzkoušet a následně provést test, jestli bylo vše správně pochopeno. Toto pojetí vede k rozdělní účelu aplikace a vytvoření logických celků, které budou podrobněji rozebrány v následujících kapitolách.

3.1 Analýza

3.1.1 Rozdělení aplikace na logické celky

Zadaná bakalářská práce, výukový program pro demonstraci principů barev a barevných modelů, by měla nabízet možnost dozvědět se něco více o teorii barev a barevných modelech. Dále jako v každé výukové aplikaci je potřeba nabídnout uživateli, studentům, možnost si znalosti prakticky vyzkoušet na modelech. Jako poslední logický celek pak poslouží testovací část, kde si studenti budou moci jednoduchým způsobem vyzkoušet, zda základní a nejpodstatnější pojmy správně pochopili. Aplikace je tak rozdělena do tří logických celků, které zvýší přehlednost a orientaci uživatele.

3.1.2 Modely

Sekce modelů se skládá ze 4 částí. Jedná se o převod mezi barevnými modely. Aplikace dokáže převádět mezi sebou hodnoty modelů HLS, HSV, RGB a CMY. Je tedy možné snadno převést jednu hodnotu na druhou a přitom mít náhled na právě převáděnou barvu. Stejně tak je možné namíchat barvu pomocí uživatelsky přívětivějšího modelu jako např. HLS a získat snadno hodnotu RGB např. pro tvorbu webových stránek.

Další model umožní demonstrovat rozklad připravené fotografie na barevné složky modelu RGB. Dále by v této části aplikace mělo být umožněno předvést změnu jasu, kontrastu a některé grafické efekty, např. takzvané zobrazení „sepia“, která připomíná starou fotografii. Vhodné by bylo i předvést některé metody filtrování barevných složek, aby bylo možno demonstrovat co nastane, pokud některou složku nebo její část nezobrazíme.

Třetí model by měl obsahovat možnosti načítání obrázků z disku uživatele a provést zobrazení histogramů jednotlivých RGB složek do přehledných grafů, aby bylo možno prezentovat jejich složení u různých typů obrázků.

Následující a poslední model nabídne 3D zobrazení jednotlivých barevných modelů. Měl by umožnit představení rozložení barev v modelu, jejich vzájemné propojení a pomocí interaktivního ovládání zajistit, aby si uživatel mohl lehce vyzkoušet co se stane, pokud pohne určitou složkou v daném modelu a jak to pozmění nastavovanou barvu.

3.1.3 Výuka

Vhodné řešení výukové části, která by umožnila dobrou čitelnost a zároveň přehlednost, je použití integrovaného HTML prohlížeče. Formou jednoduchých HTML stránek budou prezentovány základní znalosti nutné pro pochopení dané problematiky. Každé podkapitole barev a barevných modelů bude věnována jedna samostatná HTML stránka. Pro zajištění maximální kompatibility bude vhodné použít pouze technologii značkovacího jazyka HTML v kombinaci s kaskádovými styly CSS.

3.1.4 Testy

Testovací část by měla nabízet možnost určité výuky pomocí testů a testování pro ověření znalostí problematiky. V rámci procházení testových otázek se také nabízí možnost zahrnout do programu určitá vysvětlení správné odpovědi každé otázky. Stejně tak musí být zajištěny standardní testy, pomocí kterých se uživatel – student dozví, jak danou oblast zvládl. Stojí za úvahu promyslet systém, u kterého lze otázky v případě potřeby aktualizovat a to bez potřeby úpravy vlastního programu.

3.2 Návrh řešení

V této kapitole spojíme některé teoretické návrhy z oblasti analýzy s budoucí implementací dané části aplikace. Vybereme operační systém, pro který bude naše aplikace určena, vhodné vývojové prostředky a význam důležitých metod a tříd, které bude v aplikaci vhodné použít.

3.2.1 Volba OS a vývojových prostředků

Cílová skupina výukového programu zpracovávaného v rámci této bakalářské práce jsou studenti se zájmem o počítačovou grafiku. Jako programovací jazyk byl zvolen C# v kombinaci s vývojovým prostředím MS Visual Studio .NET 2005. K tomuto systému je možné získat akademickou licenci MSDNAA a tak bezplatně využívat tento rozsáhlý vývojový systém. Záměrně byla zvolena starší verze tohoto systému – v době psaní práce byla již dostupná verze 2008 – z důvodu, že by pro účely této aplikace nebylo vhodné použití novější knihovny .NET než verze 2.0, a to z důvodu zpětné

kompatibility. Vlastnosti přidané do novější verze knihovny .NET by tak nebyli využity. V době psaní programu byla aktuální verze 3.5, ale jak už je uvedeno výše, zhoršila by se tím zpětná kompatibilita se staršími verzemi operačních systémů firmy Microsoft. Aby bylo možné program úspěšně spustit, je potřeba zajistit, aby v systému byla nainstalována správná verze této knihovny, která je však nativně podporována až v nejnovějších operačních systémech. Bylo by tak třeba zajistit společně s aplikací distribuci knihovny .NET, která však řádově desetkrát převyšuje velikost celé aplikace. Po vyhodnocení a započítání těchto faktorů byla zvolena knihovna .NET verze 2.0. Pro zpracování 3D modelu v aplikaci bylo rozhodnuto nevyužít Direct3D nebo různých Frameworků pro podporu OpenGL v prostředí .NETu, ale po prozkoumání složitosti komerčních řešení a chybovosti open-source nástaveb pro C#, bylo jako vhodné řešení zvoleno naprogramovat 3D model v aplikaci jako samostatný malý program pomocí C/C++ jen za využití knihovny GLUT[9]. Tento model by byl v aplikaci volán po stisku tlačítka a spuštěn jako zvláštní aplikace. Běžný uživatel by tedy nemusel postřehnout, že model není naprogramován jako součást hlavního programu.

Volbou vývojových prostředků je dána i volba operačního systému. Aplikace podporuje systémy schopné nainstalovat knihovnu .NET verze 2.0, což jsou v podstatě všechny dnes používané systémy z této řady Windows. Výčtem jde tedy o Windows 98SE, Windows ME, Windows 2000 SP4 a novější, Windows XP SP2 a novější, Windows Vista 32/64b. Podle předpokladů by měla být bezproblémová funkčnost i v systému Windows 7, který byl v době vývoje aplikace k dispozici pouze jako beta-verze. Ve všech těchto systémech bude funkčnost aplikace ověřena – toto bude popsáno dále v sekci testování.

Dokumentace zdrojového kódu je zajištěna pomocí zápisu komentářů v XML tvaru přímo do zdrojového textu, které umí zpracovat jak samotné Visual Studio, tak i programy určené pro tvorbu nápovědy a dokumentací jako je např. Doxygen.

3.2.2 Návrh řešení logických celků a použití knihoven

Na obrázku (schéma 3.1) je přehledně znázorněno logické rozvrstvení jednotlivých částí v aplikaci tak, jak bylo popsáno výše v analýze řešení. Následující text v této kapitole se zaměřuje na použití různých řešení, knihoven a technologií vhodných pro implementaci dané části aplikace.

V této podkapitole bude text zaměřen na využití vhodných open source knihoven třetích stran, které by umožnily lepší a rychlejší vývoj aplikace. Nejde zde o to maximálně si práci ulehčit, ale použít funkce metody na danou činnost optimalizované. Po zvážení návrhu byla do aplikace zahrnuta knihovna AForge.NET [10]. AForge.NET je knihovna obsahující funkce a metody, které nám umožní předvést v aplikaci více funkcí a efektů, než by bylo jinak možné a ukázat tak maximum co z oblasti barev barevných modelů lze.

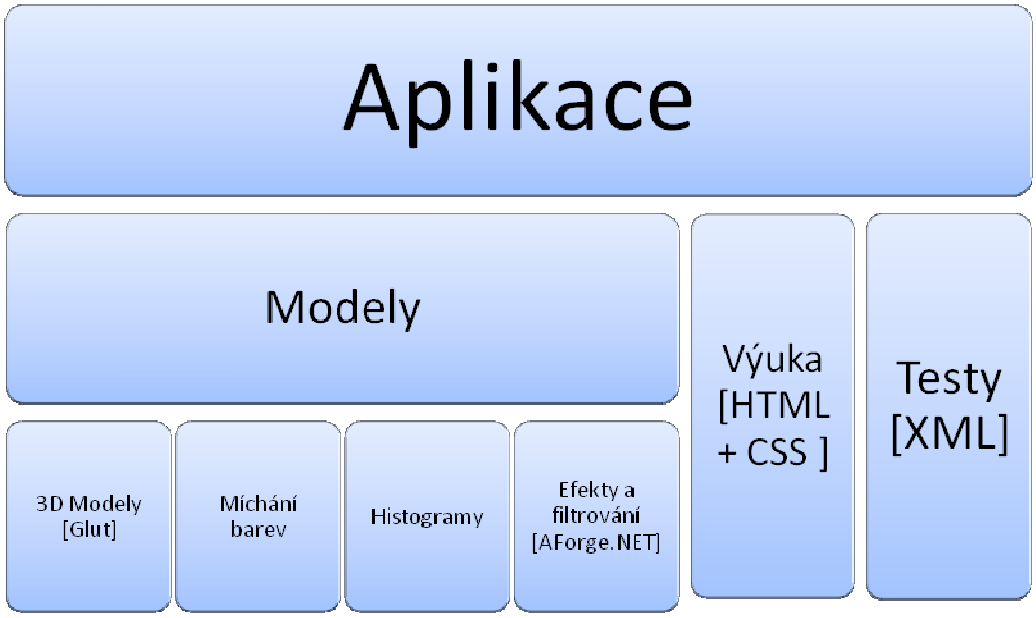


Schéma 3.1 - Logické celky aplikace

Další logický celek aplikace, který je třeba vhodně navrhnout, je část testů. Zde se přímo nabízí použití databáze, ovšem pro účely aplikace, kde bude použito maximálně pár desítek testovacích otázek je systém MS-SQL příliš velký a monstrózní. Nebylo by také tak snadné zajistit úpravu testovacích otázek bez části aplikace, která by se updatem, vkládáním a mazáním otázek zabývala. Z tohoto důvodu by mohlo být ideální řešení použití XML souboru pro strukturované a značkované uložení otázek a to tak, aby se s nimi dalo pracovat v aplikaci jako s databází a zároveň byly otázky snadno citovatelné v jakémkoliv textovém editoru. Načtení do aplikace by bylo řešeno pomocí objektu *dataset* a jeho metody *readXml()*. Tyto vlastnosti by zajistili práci s daty jako databází, ovšem načítala by se z XML souboru. Tím by byly zajištěny pozdější možné úpravy testů bez nutnosti program znovu překládat.

Výukovou část není třeba nějak podrobněji rozebírat nebo navrhovat. V aplikaci využije vestavěné komponenty, která zprostředkovává služby jádra webového prohlížeče. Tímto nám umožní snadno zobrazit v aplikaci výukovou část bez nutnosti programovat si vlastní prohlížeč. Protože aplikace si bere jako prohlížeč vždy ten, který je v systému nainstalován, bylo by vhodné se v této části držet jen základní grafiky.Zajistit přehlednost textu a dobrou čitelnost a nepouštět se do rozsáhlejšího ladění vzhledu z důvodu, že uživatel může mít ve svém systému prohlížeč naše funkce nepodporující. Tímto by práce přišla nazmar a ještě by bylo znemožněno program používat.

Závěrečnou, ale rozhodně významnou část aplikace bude tvořit sekce modelů. Tato část aplikace bude obsahovat čtyři modely. První z nich nám reprezentuje jakousi barevnou „kalkulačku“. Dovolí převádět hodnoty z jednoho barevného modelu na všechny (RGB, CMY, HSV, HLS) ostatní a zároveň převáděnou barvu zobrazit. Tato část bude v reprezentaci barevných modelů do určité míry

souviset s 3D modelem v aplikaci. V něm se bude zobrazovat barevný model v 3D prostoru a zvolená barva se zobrazí v modelu jako „kulička“ na příslušné pozici, obarvená na odpovídající barvu. Při přepnutí na jiný model bude nastavení zachováno a „kulička“ se zobrazí na pozici, jaká odpovídá stejné barvě po převodu z jednoho modelu na druhý. Umožní nám to tedy pohnout jednou složkou v zkoumaném modelu, přepnout se do jiného a zjistit, jak se přesunula hodnota v modelu druhém. Tato interaktivní změna by měla poskytnout uživateli dostatek vyžití při zkoumání těchto barevných modelů.

Jak už bylo popsáno výše, pro ukázky v části o efektech a filtrovaní bude použita knihovna AForge.NET [10], která je velmi dobře optimalizovaná pro práci s obrázky, grafickými efekty ale i matematikou. Jedná se o velice kvalitní knihovnu jejíž metody svojí rychlostí překonávají vestavěné funkce knihovny .NET, konkrétně jejího „namespace“ zabývajícího se prací s grafikou. Tato vlastnost je i pouhým okem patrná při porovnání této části aplikace s posledním modelem a to histogramem. Zde nebude potřeba rychlé změny a přepočítávání obrazu při nastavování hodnot pomocí uživatelského rozhraní, a tak zde bude využito standardních metod objektů *Bitmap* a *Pixel* z knihovny .NET.

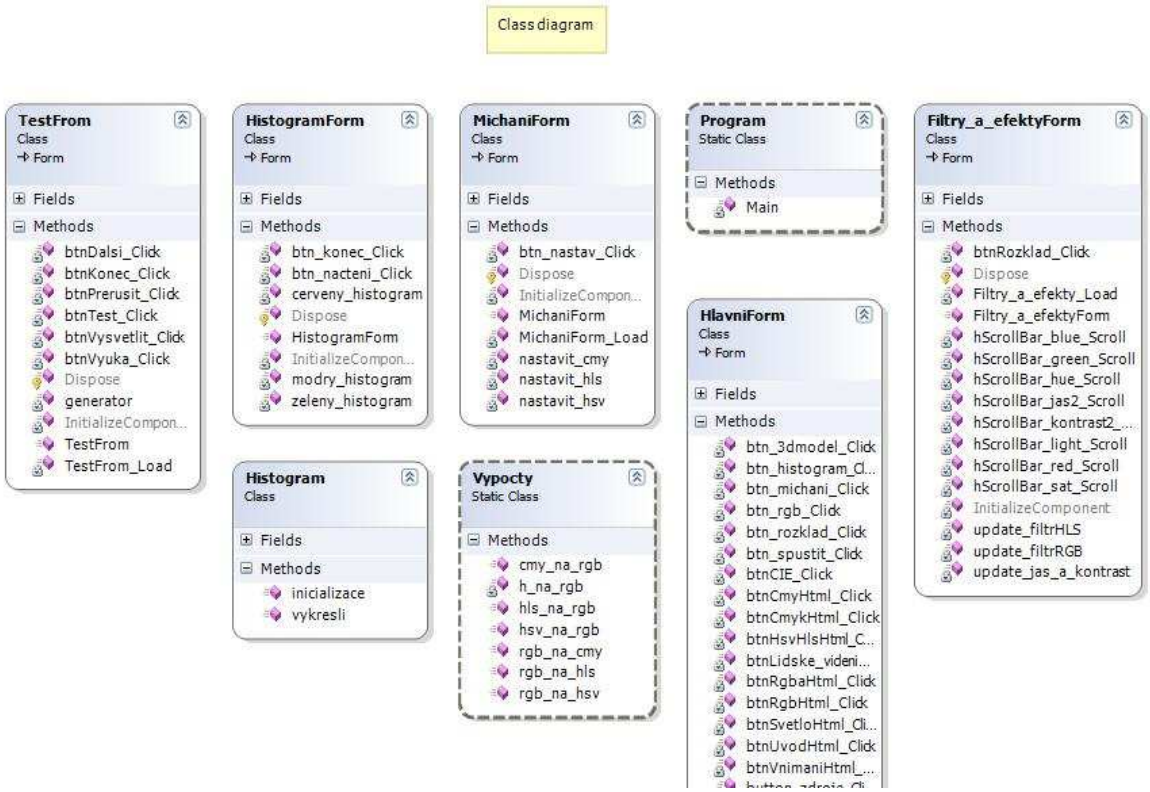
3.2.3 Význam tříd

V této podkapitole bude popsán základní význam jednotlivých tříd v aplikaci. Grafický přehled jednotlivých tříd přehledně zobrazuje obrázek 3.2, na kterém kromě přehledu tříd můžeme vidět jaké metody budou jednotlivé objekty daných tříd poskytovat. Základní třídou, která umožní spuštění, je *static class Program*, která má jako hlavní úkol vytvoření objektu třídy *HlavniForm*. Třídy s názvem "*Form" značí třídu, která zdědila vizuální objekty třídy *Form*. Třída *HlavniForm* je nejdůležitější třídou aplikace, která vizuálně spojuje všechny její části. Obsahuje objekt zastupující internetový prohlížeč umožňující zobrazení výukové části aplikace. Dále pak tato třída umožní přes svoje vizuální komponenty spuštění další tříd – např. *MichaniForm*. Tato třída bude mít na starosti zobrazení namíchaných barev a ovládání přepočtu v různých modelech. K tomuto účelu využívá třídu *Vypocty*, což je statická třída obsahující funkce pro převod mezi jednotlivými barevnými modely.

Další třídou je třída *Filtry_a_EfektyForm* , která obsahuje metody prezentující grafické efekty, rozklady na barevné složky a filtrování jednotlivých složek. Využívá k tomu knihovny AForge.NET [10], která bude dále popsána v části implementace.

Třída *HistogramForm* vizuálně zapouzdřuje objekty třídy *Histogram*. Ty budou umožňovat zpracování histogramu ze zadaného obrázku a histogram vracet zpět jako rastrový obrázek.

Třída *TestForm* v aplikaci reprezentuje testovací část aplikace. Nabízí možnost testů, výukového režimu a práce s XML soubory.



Obrázek 3.2 - Přehled tříd v aplikaci

Jak již bylo napsáno v kapitole 3.2.1, největší problém nastal s návrhem a následnou implementací 3D modelů v aplikaci. Aplikace a je navržená v systému .NET za použití jazyka C#. Při návrhu 3D modelů však vyšlo najevo, že tato platforma není vhodná pro využití knihovny OpenGL resp. takzvané Open-source knihovny a veřejné Frameworky pro C# se ukázaly velmi nekvalitní a pomalé. Některé základy funkce knihovny OpenGL nebyly vůbec implementovány nebo způsobovaly pády systému. Řešení této situace si vynutilo změnu návrhu, a proto jsou zmíněna již zde. Nejjednodušší řešení by bylo modely úplně vynechat, ale potom by aplikace ztratila podstatnou část ze svých demonstračních schopností, proto bylo toto řešení zavrhnuto. Dále pak bylo možné použít některých velkých placených knihoven pro grafiku v C#. Tyto knihovny kompenzují určitou pomalost grafiky v C# proti C/C++ ovšem za cenu své velikosti a hlavně jsou to drahé a placené nástroje. Z tohoto důvodu bylo zvoleno kompromisní řešení, kdy 3D modely v aplikaci jsou navrženy jako externí aplikace za pomoci jazyka C/C++ a knihovny GLUT[9]. V aplikaci jsou spouštěny pomocí externích procesů a byla snaha o co největší skrytí toho, že aplikace není jeden celek. Běžný uživatel by tak tento rozdíl nemusel poznat. Vlastní implementace této části bude popsána v následující kapitole.

4 Implementace

V této kapitole bude popsána implementace některých důležitých metod, které mají zásadní význam pro základní funkci aplikace. Bude zde rozebráno, jak bylo implementováno grafické rozhraní, jednotlivé podstatné části aplikace a 3D modely za použití OpenGL[11] a dalších podpůrných knihoven. V závěru kapitoly v sekci testování bude uvedeno, jak byla aplikace odzkoušena, na jakých operačních systémech spolehlivě funguje a jaké jsou její hardwarové nároky. Nakonec bude představeno několik obrázků z běhu aplikace a malý komentář k jejímu ovládání.

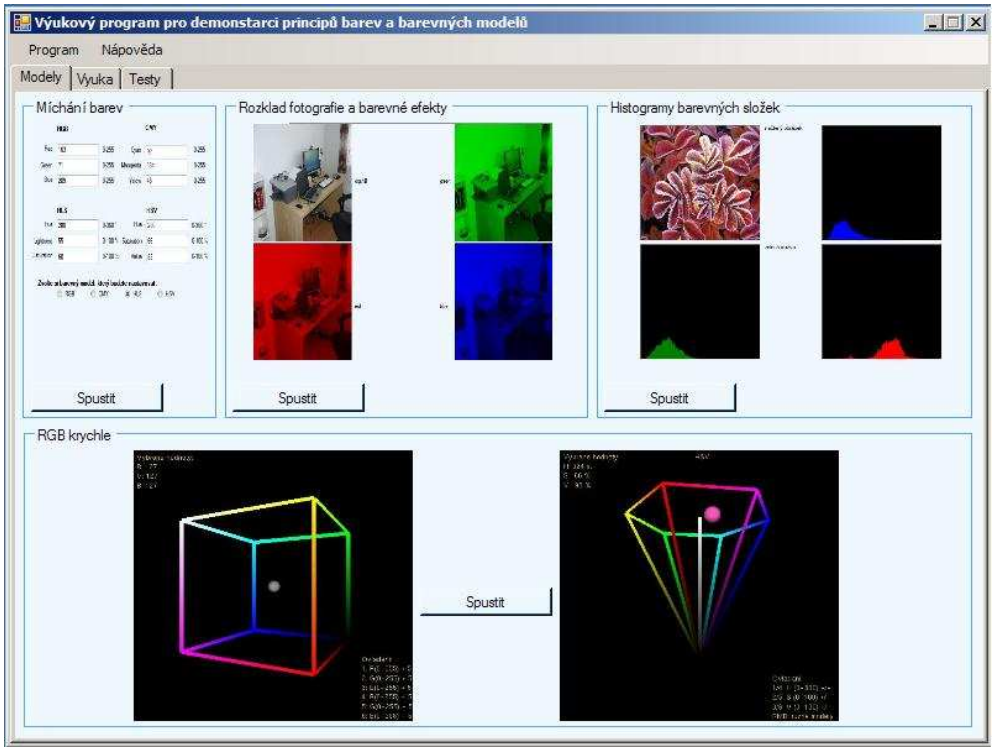
4.1 Grafické rozhraní

Grafické rozhraní aplikace by podle analýzy mělo zajišťovat rozdělení na logické celky,a to na část modelů, výukovou část a testovací část. Tohoto je docíleno pomocí objektu třídy *System.Windows.Forms.TabContorl* z knihovny .NET 2.0, který aplikaci umožnil přirozené rozdělení na tři části a to pomocí pro typických záložek. Toto řešení bylo zvoleno z důvodu přehlednosti. Rozlišení pro aplikaci bylo napevno nastaveno na 800, 600px z důvodu dobré ergonomie aplikace i při použití na stále oblíbenějších mini-noteboocích, který mají nativní rozlišení pouze 1024 na 600px. Obrázek 4.1 naznačuje grafický vzhled aplikace vytvořený pomocí funkcí knihovny .NET.

Aby byla zajištěna dostatečná plocha pro vlastní modely, je hlavní aplikace jen prostředek ke spuštění vlastních aplikací pro práci s modely. Ty jsou v aplikaci celkem čtyři. Spouští se z hlavní aplikace reprezentované třídou *HlavniForm*, která dědí vlastnosti třídy *System.Windows.Forms*, pomocí tlačítek opět z knihovny .NET 2.0. Tři jsou 2D modely, které si vystačí se základním rozhraním GDI (Graphics Driver Interface) a jeden, nebo chcete-li čtyři v jednom, je řešen jako 3D model pomocí grafické knihovny OpenGL[11]. Více o těchto modelech bude uvedeno v následujících podkapitolách.

4.1.1 2D modely v aplikaci

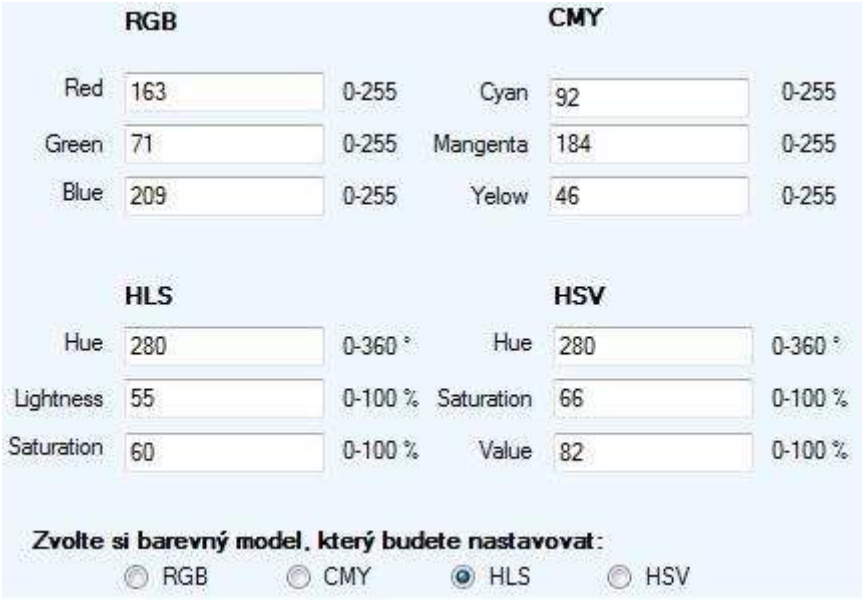
Tyto modely v aplikaci slouží hlavně k tomu, aby si studenti prakticky vyzkoušeli práci s barvami, jejich míchání, převádění mezi barevnými modely, filtrování, různé efekty a úpravy. Informace k vlastní implementaci těchto modelů jsou uvedeny v následujících podkapitolách.



Obrázek 4.1 - Hlavní aplikace

4.1.1.1 Míchání barev

Tento model je reprezentován třídou *MichaniForm*, která dědí ze třídy *System.Windows.Forms*. Na ploše formuláře jsou komponenty umožňující zadávání hodnot v různých barevných modelech. Následně musí uživatel zvolit, jaký model nastavoval, resp. podle kterého se budou nastavovat ostatní. Po stisku tlačítka se provede přepočítání na ostatní barevné modely.



Obrázek 4.2 - Míchání barev

Vlastní přepočítání se děje pomocí metod třídy *Vypocty* , které umožňují převod jakéhokoliv ze čtyř modelů RGB, CMY, HSV a HLS na model RGB a zase zpět. Tato třída je na implementaci aplikační logiky nezávislá a slouží jen pro získání převáděné hodnoty. Lze ji tedy snadno použít i jinde.
K vlastní implementaci této třídy. Tato třída nedědí z žádné standardní třídy knihovny .NET. Třída obsahuje public static metody umožňující vlastní převod. Postup pro převod mezi jednotlivými modely má charakter algoritmu a je možné ho získat na odborných webových stránkách zabývajících se problematikou barev nebo v publikaci jako např. [13], ve které lze najít vše potřebné. Ve třídě je implementováno šest hlavních metod:

public static void rgb_na_cmy(double r, double g, double b, out double c, out double m, out double y)
public static void cmy_na_rgb(double c, double m, double y, out double r, out double g, out double b)
public static void rgb_na_hls(double r, double g, double b, out double h, out double l, out double s)
public static void hls_na_rgb(double h, double l, double s, out double r, out double g, out double b)
public static void hsv_na_rgb(double h, double s, double v, out double r, out double g, out double b)
public static void rgb_na_hsv(double r, double g, double b, out double h, out double s, out double v)

Význam je zřejmý již z názvu a vstupních - výstupních parametrů metod. Metody zajišťují prostý převod hodnot bez závislosti na grafickém rozhraní. V rámci tohoto modelu jsou implementovány i základních funkce pro ochranu před zadáváním nevhodných parametrů, např. zadáním hodnoty barevné složky pomocí slova (třicet) namísto čísla. Je to ochrana pouze základní, pro nasazení aplikace do běžného užívání by bylo potřeba ošetřit všechny možné kombinace chyb, které by mohl uživatel vymyslet. Obrázek 4.2 dokládá, pro lepší představu, reálný vzhled této části aplikace.

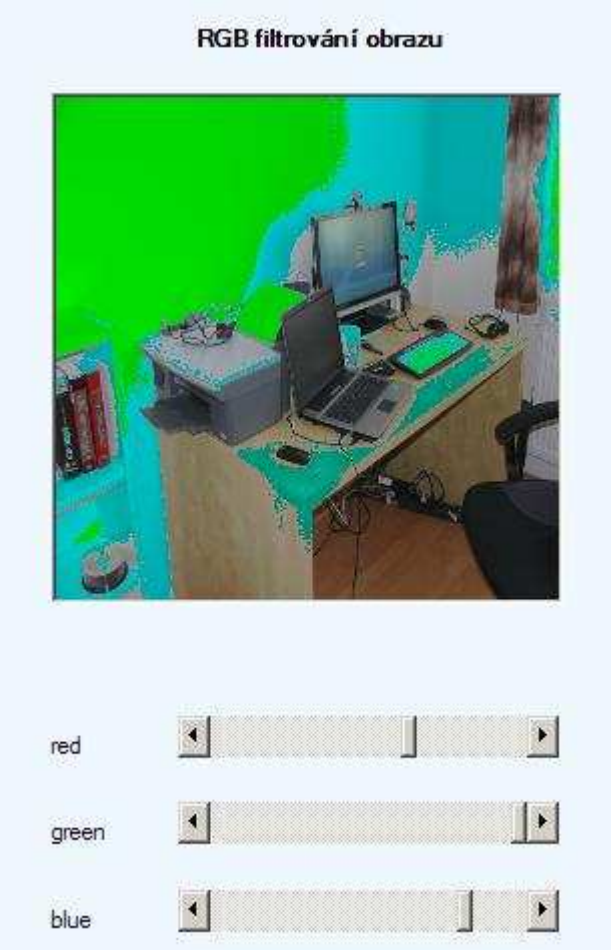
4.1.1.2 Filtrování a efekty

Model pro demonstraci grafických efektů s barvami a jejich filtrováním, je v aplikaci reprezentován třídou *Filtry_a_efekty*, která dědí ze třídy *System.Windows.Forms*. Aplikace je standardní komponentou ze třídy *System.Windows.Forms.TabControl* rozdělena na tři části. Konkrétně na rozklad fotografie na barevné složky RGB, dále pak úpravy složek filtrováním a to jak RGB složek, tak filtrováním HSV hodnot a nakonec grafické efekty a intenzity jasu a kontrastu.

K předvedení těchto grafických efektů bylo použito knihovny AForge.NET[10] , která nám umožnila předvést více grafických efektů než by bylo možné udělat při ručním zpracování. Standardní funkce pro práci s grafikou GDI+ ve VS.NET jsou totiž příliš pomalé a opravdu se vyplatí je nahradit nějakou kvalitní open-source knihovnou jakou AForge.NET bezesporu je. Nejedná se tady o maximální si ulehčení práce, ale o možnost studentům ukázat co nejvíce z dané oblasti a soustředit

se na to podstatné, ukázat maximum co jde a vytvořit co nejlepší výukový program. Není zde úkolem předvést, co vše jde k dané problematice naprogramovat.

Aplikace pomocí funkcí výše uvedené knihovny rozloží obrázek na základní složky modelu RGB. V další části uživateli umožní prozkoumat, co se stane s obrázkem pokud vyfiltrujeme z obrazu část barevné složky, a dovolí uživateli měnit hodnotu jasu a kontrastu na obrázku. Vzhledem k zaměření aplikace lze za nejdůležitější část tohoto modelu považovat filtrování barev v obrázku. Tato úprava je zajištěna pomocí objektu *filtr* třídy *ChannelFiltering*, který je napojen na grafické rozhraní aplikace. Při změně polohy *scrollbaru* je zavolána metoda *update_filtrRGB()* a vyfiltrovaný obrázek je zneplatněn. Je načtena nová kopie původního obrázku, znovu nastaven filtr a použit na kopii originálního obrázku. Vzhledem k rychlosti funkcí knihovny AForge.NET toto není problém, na rozdíl od změny pomocí systémových funkcí .NETu *SetPixel* a *GetPixel* , které jsou bohužel velmi pomalé.



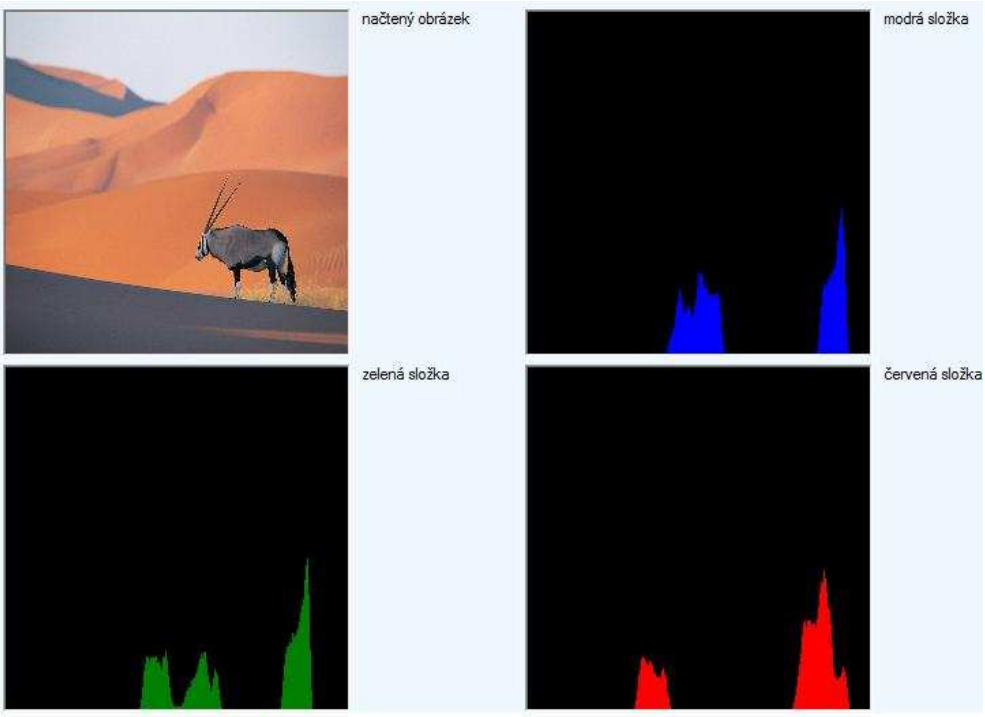
Obrázek 4.3 - RGB filtrování

Záměrně zde nebylo použito změny barevné složky ve smyslu zvýšení resp. snížení její hodnoty, ale skutečně vyfiltrování. Pokud tedy nastavíte "posuvník" na hodnotu 128, pak každý pixel obsahující hodnotu dané složky větší než 128 bude mít tuto složku nastavenu na 0. Dojde tedy k jejímu vyfiltrování z obrazu. Toto umožňuje pro mnohé zajímavé efekty. Na bílé stěně se změna projeví i při

minimálním posunu. Složky všech pixelů jsou tu blízko maximální hodnoty 255, kdežto na tmavé podlaze se změna vůbec neprojeví (obr. 4.3). Principielně stejně jako u tohoto filtrování bylo postupováno i u dalších, jako je filtrování jasu, kontrastu a tónu.

4.1.1.3 Histogramy

Tento jednoduchý, ale zajímavý model je implementován pomocí standardních funkcí GDI knihovny obsažené v "balíku" .NETu. V tomto případě nedochází k interaktivním změnám, takže nám jeho rychlost bude dostačovat. Vzhledem k tomu, že v tomto modelu se nachází tři histogramy, každý pro jinou barevnou složku, bylo zvoleno řešení implementovat samostatnou třídu Histogram. Tato třída implementuje metodu, která umožňuje předat objektu této třídy obrázek a jako návratovou hodnotu obdržet bitmapu s histogramem dané složky z obrázku.



Obrázek 4.4 - Histogram

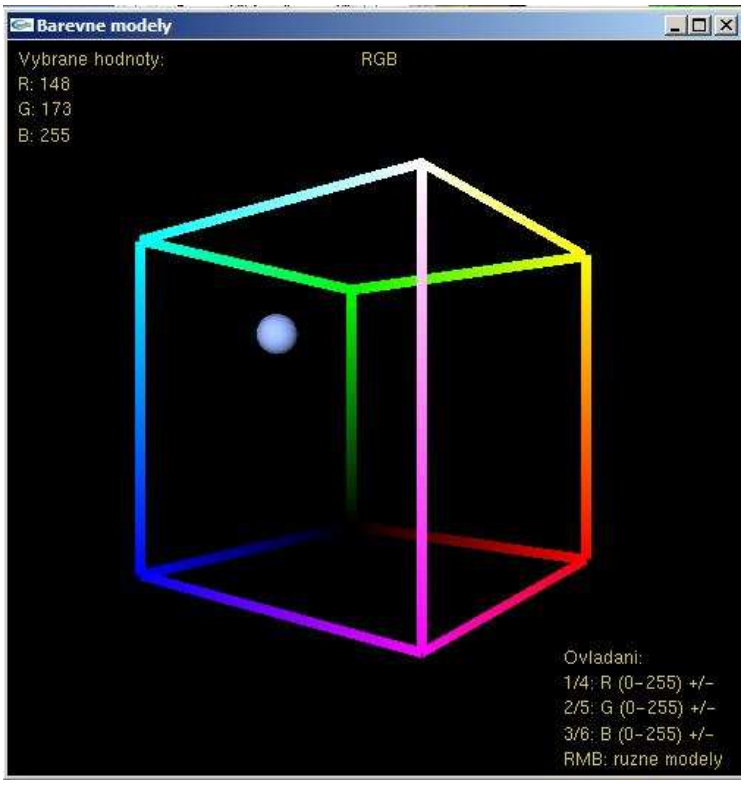
Obrázek i histogramy jsou v aplikaci napevno nastaveny na rozměr 256x256 obrazových bodů. K vlastní tvorbě bitmapy histogramu. Objekt třídy *Histogram* obsahuje celočíselné pole o rozměru 256 prvků, které je po inicializaci nastaveno na hodnotu 0. Toto pole představuje vlastní histogram. Obrázek se prochází pixel po pixelu a pomocí funkce GDI *GetPixel(x,y)*, se zjistí hodnota dané barevné složky v pixelu. Do daného prvku pole, který odpovídá této hodnotě, se pak přičte jednička za pixel s touto hodnotou. Po projití celého obrázku tam máme pole jehož 256 prvků reprezentuje rozsah dané barevné složky, tedy 0-255 a v každé této hodnotě je obsažen počet pixelů s těmito

hodnotami, které se v obrázku nacházejí. Po dokončení této fáze se přistoupí k vykreslování do bitmapy. Za každých 20 pixelů s danou hodnotou složky v poli se vykreslí do bitmapy jeden barevný bod. Pokud i tak jich je přes 256, budou ostatní ignorovány, protože by se už nevešli na obraz. Změna měřítka by zase způsobila přílišnou redukci ostatních hodnot v histogramu.

4.1.2 3D modely v aplikaci

Jak již bylo uvedeno v části návrhu, tento model není naimplementován v jazyce C#, ale jako samostatný program v jazyce C++. Důvody proč tomu tak je, byly osvětleny v kapitolách 3.2.1 a 3.2.3. Zde bude věnován prostor základním implementačním metodám, které byly pro tento model použity. Aplikace je navržena pomocí knihovny GLUT a je naprogramována za použití procedurálního paradigmatu. Při práci na této části modelu byla velmi užitečná kniha o programování za použití knihovny OpenGL[12]. Základním kamenem v aplikaci jsou tedy funkce. Vlastní program je rozdělen do několika zdrojových souborů. Z hlediska logiky aplikace na to ale není třeba brát zřetel. Aplikace využívá napevno stanoveného rozměru okna zvoleného tak, že nebrání v použití ani na zařízeních s nižším rozlišením. Vzhled modelu je zobrazen na obrázku 4.5.

Základní funkce pro tento model jsou *callback_display*, *callback_keyboard*, *callback_menu*, *graphic_text* a vlastní funkce pro kreslení modelů: *scene_rgb*, *scene_cmy*, *scene_hsv*, *scene_hls*. Vlastní výkonný kód je komentován ve zdrojovém kódu a tyto komentáře jsou upraveny do programové dokumentace, které je v příloze (na DVD) této práce.



Obrázek 4.5 – 3D model

Základní princip funkce aplikace je, že po spuštění programu vytvoříme okno, nastavíme pozadí a zaregistrujeme události. Tím je myšleno stisk kláves, kterými se aplikace ovládá nebo myši, pomocí které se přepínají modely. Aplikaci oživíme a již dochází k volání takzvané display funkce. Ta zjistí, kolik milisekund uběhlo od minulého snímku. Následně dojde k vymazání obrazovky, vypsání textu s popisky a nastavenými hodnotami na obrazovku.Podle toho, jaký zrovna používáme model, se zavolá přes strukturu t_model adekvátní funkce pro vykreslení daného modelu. Ta využije vypočtený úhel natočení a vykreslí krychli, jehlan či kužel s příslušnou kuličkou (ukazatelem, kde se v modelu pohybujeme). Dojde k vykreslení připraveného snímku na obrazovku a tím je tento proces ukončen.

Kromě běžného trvalého překreslování rotujícího modelu, dojde k tomuto procesu i pokud nastavené funkce zachytí stisk klávesy, po kterém by se měla kulička, ukazatel, v modelu pohnout, případně by mělo dojít k přepnutí na jiný model. Během přepínání mezi jednotlivými modely dojde k uložení zvolené barvy a při přepnutí na jiný model dojde k přepočtení této hodnoty. K tomu slouží funkce pro vzájemný převod hodnot RGB, CMY, HSV a HLS.

Podrobný popis všech nastavení a funkcí OpenGL použitých v tomto modelu je, jak už bylo uvedeno výše, uveden v programové dokumentaci.

4.2 Výuková a testovací sekce

V této části textu se zaměříme na implementaci toho nejdůležitějšího, co odlišuje aplikace zaměřující se na výuku od těch, co mají předvést, co vše lze naprogramovat a čeho je technika schopna. Hlavním důvodem implementace těchto částí je zajistit co možná nejkomplexnější systém výuky pro danou oblast. Od vyzkoušení si modelu, přes teorii až po otestování znalostí.

4.2.1 Výuka

Jak již vyplynulo z analýzy, pro implementaci této části aplikace bylo nejvhodnější zvolit formu integrovaného internetového prohlížeče, který by umožnit zobrazení HTML stránek v rámci aplikace. Za tímto účelem byla v aplikaci použita komponenta *webBrowser* ze standardní knihovny. Ta využívá pro zobrazení stránek jádro *Trident API*, které je neoddělitelnou součástí desktopových systémů firmy Microsoft, bez ohledu na to, jestli v budoucnu bude možné prohlížeč Internet Explorer ze systému odstranit, či nikoliv.

Použité technologie pro stránky s výukovými texty jsou značkovací jazyk HTML 4.01 Transitional a jednoduché kaskádové styly CSS. Jako základní a nejdůležitější vlastnost při tvorbě stránek byla jejich dobrá čitelnost a přehlednost. Texty byly rozsahem voleny spíše kratší, z důvodu

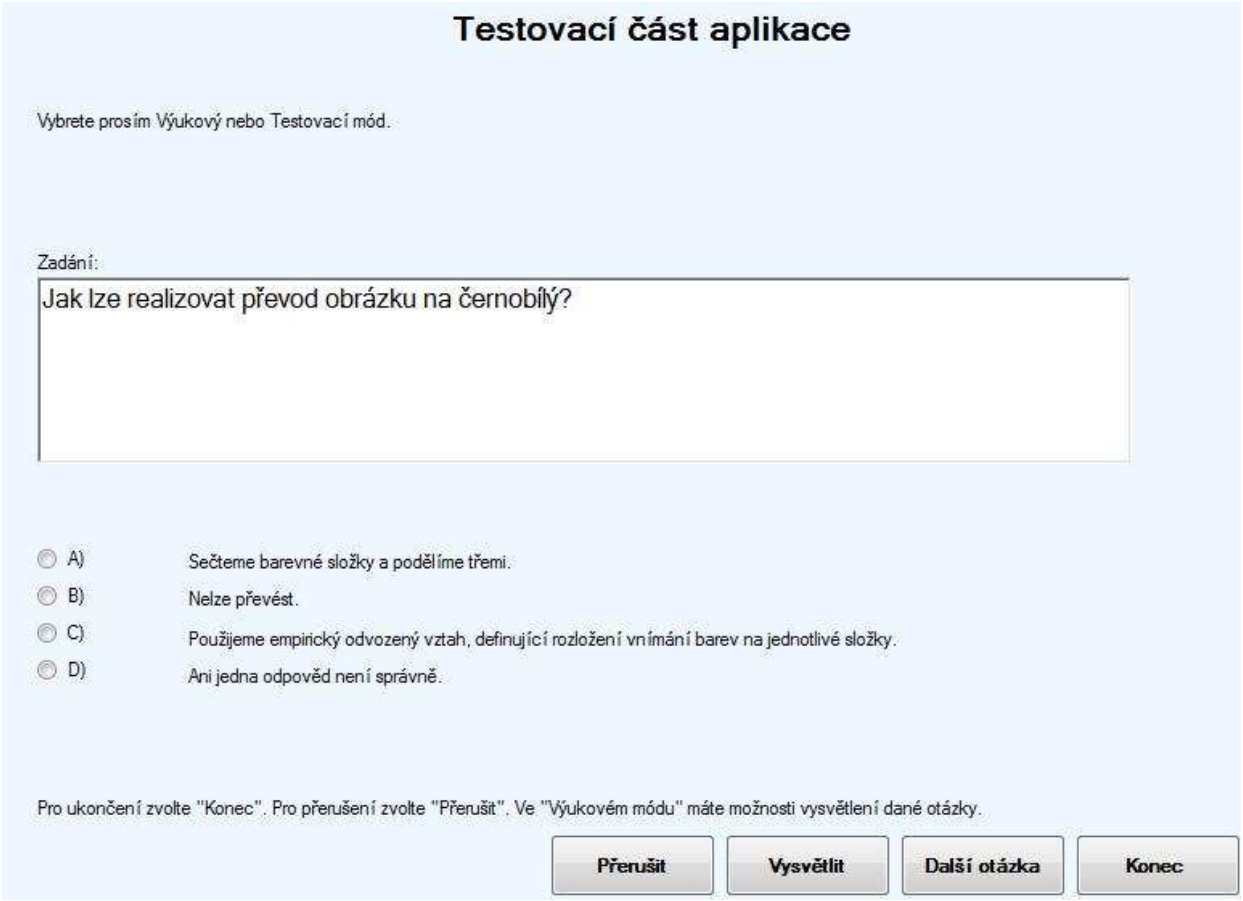
horší čitelnosti textu z obrazovky počítače. Byla zde snaha dostat do prostorově omezených textů co nejvíce základních informací, ale i zajímavostí, které jsou schopny zvýšit čitelnost textu a zvýšit zájem studentů o toto téma.

Z hlediska možnosti následné úpravy textů ve výukové části je tento model výhodný pro svou jednoduchost a možnost aktualizací, bez nutnosti znovu kompilovat aplikaci. Stejně tak je možné použít případně tyto stránky jako samostatnou část a zveřejnit na webu, případně rovnou umístit stránky na server a nastavit komponentu *webBrowser* na načítání z určitých URL adres. Aplikace bude fungovat stejně, jen by bylo nutno změnit směrování u navigačních prvků. Toto řešení by však vyžadovalo od uživatele připojení k internetu. V rámci této bakalářské práce je tedy implementováno lokální umístění stránek na disku a jejich distribuce spolu s aplikací.

4.2.2 Testy

V této části aplikace byl v průběhu analýzy hlavní požadavek na to, aby si studenti - uživatelé, mohli ověřit své znalosti na otázkách vztahujících se k danému tématu. Tuto činnost v aplikaci obstarává třída *TestForm*, která implementuje vzhled této části aplikace a zajišťuje propojení vizuálních komponent s daty (obrázek 4.6). Ta byla původně ve formě databázového souboru, ovšem tento model by neumožnil jednoduchou dodatečnou úpravu otázek bez speciálního softwaru nebo překompilování programu, bylo proto zvoleno řešení, kdy data (zadání otázek a jejich odpovědi) jsou uložena v otevřeném formátu XML, který umožní snadnou editaci bez kompilace a použití komerčních nástrojů. Vzhled testovací části se skládá z textového pole odvozeného od standardní třídy *richTextBox*, do kterého se načítá zadání otázek a čtyř komponent třídy *radioButton* pro výběr správné odpovědi. Zadání odpovědí je též načítáno ze soboru XML a následně zobrazováno v popiscích třídy *label*. Uživatel má tedy na výběr ze 4 odpovědí, přičemž jedna odpověď je správná. To by mohlo být vnímáno jako příliš snadné pro reálné testy, ale úkolem této aplikace není uživatele vyděsit, nýbrž navnadit k dalšímu studiu.

Pro načítání dat z XML souboru je v aplikaci využit objekt třídy *dataGridView*, který však v aplikaci slouží jen jako jakési datové úložiště. Pomocí objektu *dataSet* a jeho metody *ReadXml* se načtou data z XML souboru do datové tabulky. Tento zdroj dat je následně připojen na výše uvedený objekt *dataGridView*. Při vybírání otázek, načítání odpovědí, zobrazení vysvětlení odpovědí a jiných operací, je pak již soubor z XML daty pro aplikaci odstíněn a pracuje se jen s daty načtenými v tomto objektu.



Obrázek 4.6 – Testovací část

Pro možnost výuky formou testů a vlastní testování jsou v aplikaci naprogramovány dva režimy funkčnosti - výukový a testovací mód. Ve výukovém módu jsou postupně podle pořadí procházeny jednotlivé otázky z XML souboru a po každé odpovědi je vyhodnoceno, zda byla odpověď správná a je možno zobrazit vysvětlení proč je daná odpověď správně. Toto vysvětlení se zobrazuje ze souboru XML. Tyto informace se zobrazují formou *MessageBoxu*, které v standardní knihovně .NET reprezentují dialogovou komunikaci s uživatelem. Pokud uživatel prostuduje všechny otázky, je mu tato skutečnost oznámena a začíná se znovu. V případě, že ho tento režim testování přestane bavit, je možnost výukový mód přerušit a zvolit testovací mód aplikace.

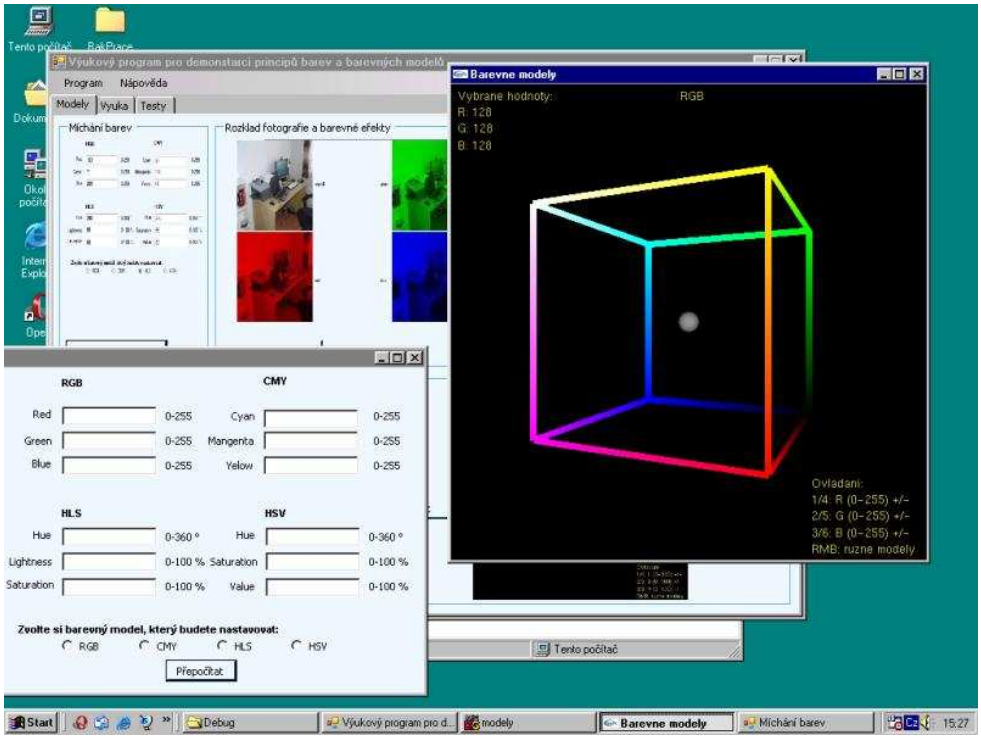
Při testovacím módu je třeba zajistit, aby se některá otázka během testu neobjevila vícekrát. Je tedy použit seznam již zobrazených otázek a generátor náhodných otázek pro jejich výběr. Ve třídě *TestForm* je pro tuto činnost použit objekt *List_otazky* třídy *List* z jmeného prostoru *System.Collections*, který zaznamenává všechny již zobrazené otázky. Pokud generátor náhodných otázek, který je implementován jako objekt třídy *Random*, vybere otázku, je porovnána s pořadovým číslem již zobrazených otázek a v případě shody je generování spuštěno znovu. Tímto způsobem je zajištěno, že se žádná otázka během testu nezobrazí dvakrát. Zobrazení výsledků je formou

procentuelní úspěšnosti v testu. Pokud například uživatel odpověděl 3x dobře a 2x špatně, je mu zobrazena úspěšnost v testu 60%.

Možnost doplnění, úpravy nebo aktualizace otázek, je díky uložení v XML souboru velmi snadná. Lze ji provádět pomoci jakéhokoliv textového editoru a bez překompilování aplikace. V aplikaci se při čtení z XML souboru data dočasně uloží v objektu *dataGridView* a procházení otázek nebo testování je tedy nezávislé na počtu otázek v XML souboru uložených. Je potřeba jen dodržet strukturu dat v XML souboru. V budoucnu je tímto směrem možné aplikaci vylepšit a ošetřit speciálními událostmi chybové hlášky při poruše struktury datového XML souboru.

4.3 Testování a ukázky z aplikace

Již z použitých technologií v aplikaci vyplývá skutečnost, že aplikace bude schopna fungovat jen na operačním systému Windows. Kromě toho předpokladu potřebuje ke svému chodu knihovnu .NET verze 2.0, která je standardně součást systému Windows od roku 2004 (verze XP SP2). Tuto knihovnu lze však doinstalovat i do starších verzí Windows. V rámci testování bylo ověřováno na jakém nejstarším systému bude aplikace fungovat. Během testování se zjistilo, že aplikace je schopna bezproblémového provozu i na počítači a systému starém 10let.



Obrázek 4.7 – Aplikace pod W98SE

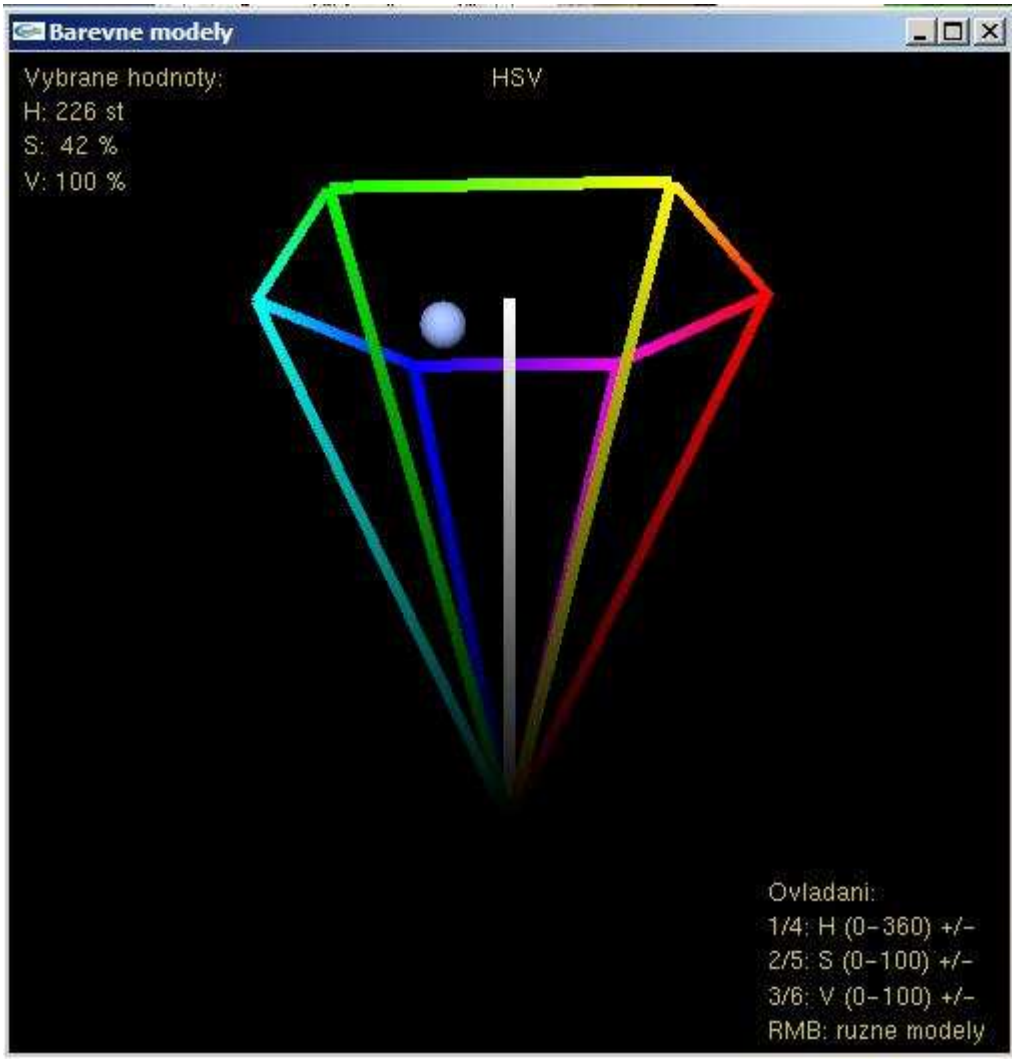
Na obrázku 4.7 můžeme vidět běh aplikace na systému Windows 98SE a bezproblémové fungování knihovny .NET na systému a hardware z konce minulého desetiletí. Aplikace úspěšně fungovala na systému s procesorem o frekvenci 550MHz a operační pamětí o velikosti 128MB. Z dnešního pohledu tedy na velmi slabém hardware. Nic by nemělo bránit provozu na jakémkoliv dnes běžně používaném počítači, notebooku nebo netbooku, splňujícím požadavky pro běh dnes používaných systémů Windows.

Z dalších systémů byla funkčnost aplikace ověřena na systému Windows XP, Vista a Windows 7Beta formou virtualizace s licencemi od MSDNAA, které jsou naší fakultě poskytovány. Testování proběhlo na stroji s těmito parametry: AMD X2 2400MHz , 2048MB RAM a VGA

adaptéru s podporou Shader Model 4.1, DirectX10.1, 120 stream procesorů, 16 texturovacích jednotek.

Následuje několik ukázek z aplikace a opravdu velmi stručný popis ovládání. Stručný proto, že ovládání celé aplikace je velmi intuitivní a jeho zvládnutí nečiní problém i méně zkušeným uživatelům, což bylo ověřeno v praxi. Samotná aplikace se ovládá stejně jako jakýkoliv jiný program, na který je uživatel ve Windows zvyklí.

Jediné místo, kde by snad nemuselo být ovládání úplně zřejmé, je ve 3D modelu (obrázek 4.8). Pro změnu barevných složek se používají číslice 1,2,3,4,5,6, což je popsáno v pravém dolním rohu. K přepnutí na jiný model dojde po použití pravého tlačítka, kdy se zobrazí kontextové menu. Z jeho nabídky si uživatel vybere model, který potřebuje. K ukončení aplikace může použít křížek vpravo nahoře nebo klávesu „ESC“.



Obrázek 4.8 – 3D model

5 Závěr

V této závěrečné kapitole budou shrnuty výsledky, význam práce, nastíněn další možný vývoj aplikace a možná vylepšení. Zadáním bakalářské práce bylo vytvořit výukový program pro demonstraci principu barev a barevných modelů. Tento program by měl sloužit jak studentům, tak i lidem se zájmem o počítačovou grafiku a problematiku barev. Během vývoje aplikace bylo usilováno o co možná největší komplexnost aplikace, tedy nesoustředit se jen na demonstrační složku, ale pohlížet na problematiku komplexněji a vytvořit skutečně výukový program.

Z hlediska základní funkčnosti je aplikace dokončena a splňuje všechny body, které byly v zadání vytyčeny. Jako hlavní přednost v aplikaci lze považovat onu komplexnost, která uživateli umožní projít celou fází studia. Tím je myšleno, že si uživatel může pročíst stručnou teorii, následně si vše prozkoušet při práci s modely a nakonec ověřit své znalosti a zkušenosti v připravených testech. Tímto se aplikace odlišuje od demonstračních programů poukazujících jen na jeden problém či téma. Modely použité v aplikaci byly vybírány s důrazem poskytnout v této problematice rozhled a zároveň vzájemné propojení barevných modelů a základních principů.

Pokud se zaměříme na další možnosti vývoje a vylepšování projektu, je zde určitě mnoho prostoru pro další práci. Především jde o zahrnutí části 3D modelů přímo do aplikace a tedy použití OpenGL knihovny přímo v prostředí jazyka C#. K tomuto by bylo vhodné provést delší fázi hledání vhodných prostředků a knihoven, protože ne všechny jsou pro tento úkol vhodné. Dále pro veřejné nasazení aplikace by bylo vhodné vytvořit instalátor aplikace, který by nabízel možnost odinstalování prostředí .NET, pokud by ho uživatel neměl k dispozici. Poslední možné vylepšení je patrné v oblasti vícejazyčnosti aplikace, kterou lze zajistit pomocí umístění veškerých textů, které se v aplikaci objevují, do externích souborů. Částečně je aplikace na tento krok již připravena. Testové otázky jsou ve formátu XML a je možné přidávat jejich verze podle zvoleného jazyka. Splněním těchto doplňujících vlastností by se program mohl stát velmi užitečnou pomůckou a vhodným výukovým a demonstračním nástrojem pro danou problematiku.

Literatura

[1] WWW stránky. Wikipedia – světlo. http://cs.wikipedia.org/wiki/Světlo. [cit. 30.12. 2008].

[2] WWW stránky. Wikipedia – barva. http://cs.wikipedia.org/wiki/Barva. [cit. 30.12. 2008].

[3] WWW stránky. Wikipedia – Elektromagnetické spektrum. http://cs.wikipedia.org/wiki/Elektromagnetické_spektrum. [cit. 28.3. 2009].

[4] Trojan S. a kolektiv.:Lékařská fyziologie. 4. vyd. Praha, Grada Publishing a.s 2003, 772s., ISBN 80-247-0512-5

[5] Žára, J., Beneš, B., Sochor, J., Felkel, P.: Moderní počítačová grafika. 2. vyd. Brno, Computer press 2004, 609s., ISBN 80-251-0454-0

[6] WWW stránky. Wikipedia – RGB. http://cs.wikipedia.org/wiki/RGB. [cit. 3.1. 2009].

[7] WWW stránky. Wikipedia – CMYK. http://cs.wikipedia.org/wiki/CMYK. [cit. 10.1. 2009].

[8] WWW stránky. Wikipedia – HLS. http://cs.wikipedia.org/wiki/HSV. [cit. 20.1. 2009].

[9] WWW stránky. GLUT – The OpenGL Utility Toolkit http://www.opengl.org/resources/libraries/glut/. [cit. 26.3. 2009].

[10] WWW stránky. AForge.NET:: Framework. http://www.aforgenet.com/framework. [cit. 26.2. 2009].

[11] WWW stránky.OpenGL. http://www.opengl.org. [cit. 2.3. 2009].

[12] Shreiner D., Woo M., Neider J., Davis T.:OpenGL – průvodce programátora. 1. vyd. Brno, Computer press 2006, 671s., ISBN 80-251-1275-6

[13] J. Foley, A. van Dam, S. Feiner, J.Hughes.: Computer Graphics – Principles and Practice. 2. vyd., Addion-Wesley , Reading, Messachusetts 1990

Seznam příloh

- DVD se zdrojovými kódy

Obsah DVD

- readme.txt - informace o adresářové struktuře disku a pokyny pro spuštění aplikace
- ovladani.pdf - popis ovládání a všech funkcí aplikace
- manual.txt - stručné informace k programové dokumentaci s odkazy na složky obsahující její kompletní verzi
- kompilace.txt - informace k překladu zdrojových kódů a informace o kompilátorech
- BakPrace.doc - elektronická verze textu bakalářské práce (i jako PDF)
- složka bin - kompletní spustitelná verze aplikace
- složka code - kompletní sada zdrojových kódů aplikace